A sampling of Randomized Algorithms

K. Subramani[1]

[1] Lane Department of Computer Science and Electrical Engineering
West Virginia University

31 January, 2012

Outline

Outline

1. Recap
2. Verifying polynomial Identities

Outline

1. Recap
2. Verifying polynomial Identities
3. Verifying Matrix Multiplication

Outline

Outline

Recap

Main points

Probability spaces, Random Variable,

Recap

Main points

Probability spaces, Random Variable, Distribution of a random variable (pmf),

Recap

Main points

Probability spaces, Random Variable, Distribution of a random variable (pmf), Expected Value,

Recap

Main points

Probability spaces, Random Variable, Distribution of a random variable (pmf), Expected Value, Variance.

Polynomial Identities and the verification problem

Polynomial Identities and the verification problem

Definition

A function $f()$ of one variable x, is called a polynomial function, if it satisfies,

$$f(x) = \sum_{i=0}^{n} a_{n-i} \cdot x^{n-i}.$$

Polynomial Identities and the verification problem

Definition

A function $f()$ of one variable x, is called a polynomial function, if it satisfies,

$$f(x) = \sum_{i=0}^{n} a_{n-i} \cdot x^{n-i}.$$

The above form of $f()$ is called the *canonical form*.

Polynomial Identities and the verification problem

Definition

A function $f()$ of one variable x, is called a polynomial function, if it satisfies,

$$f(x) = \sum_{i=0}^{n} a_{n-i} \cdot x^{n-i}.$$

The above form of $f()$ is called the *canonical form*.

Note

A univariate polynomial function can also be expressed in the form:

$$f(x) = \Pi_{i=1}^{n}(a_i x - b_i)$$

Polynomial Identities and the verification problem

Definition

A function $f()$ of one variable x, is called a polynomial function, if it satisfies,

$$f(x) = \sum_{i=0}^{n} a_{n-i} \cdot x^{n-i}.$$

The above form of $f()$ is called the *canonical form*.

Note

A univariate polynomial function can also be expressed in the form:

$$f(x) = \Pi_{i=1}^{n}(a_i x - b_i)$$

This form is called the product form.

Polynomial Representation

Representation

Polynomial Representation

Representation

Polynomial Representation

Representation

(i) Without loss of generality we will assume that the coefficient of x^n is 1 in the polynomials that we consider (both canonical form and product forms).

Polynomial Representation

Representation

(i) Without loss of generality we will assume that the coefficient of x^n is 1 in the polynomials that we consider (both canonical form and product forms).

(ii) Accordingly, a polynomial function in the canonical form can be written as:

$$f(x) = x^n + a_{n-1} \cdot x^{n-1} + a_{n-2} \cdot x^{n-2} + \ldots a_0$$

Polynomial Representation

Representation

(i) Without loss of generality we will assume that the coefficient of x^n is 1 in the polynomials that we consider (both canonical form and product forms).

(ii) Accordingly, a polynomial function in the canonical form can be written as:

$$f(x) = x^n + a_{n-1} \cdot x^{n-1} + a_{n-2} \cdot x^{n-2} + \ldots a_0$$

and a polynomial function in product form can be written as:

$$g(x) = \Pi_{i=1}^{n} (x_i - c_i)$$

Statement of Problem

Problem

Statement of Problem

Problem

Given two polynomial functions $f()$ and $g()$ in canonical form and product form respectively, is $f() = g()$?

Statement of Problem

Problem

Given two polynomial functions $f()$ and $g()$ in canonical form and product form respectively, is $f() = g()$?

Theorem

Every polynomial function has a **unique** *canonical representation.*

Deterministic Approach

Deterministic Approach

Function VERIFY-POLYNOMIAL-IDENTITY($f(), g()$)

1: Convert $g()$ into canonical form.

Deterministic Approach

Function VERIFY-POLYNOMIAL-IDENTITY$(f(),\ g())$

1: Convert $g()$ into canonical form.
2: Check if the coefficients of $g()$ match up with the coefficients of $f()$.

Deterministic Approach

Function VERIFY-POLYNOMIAL-IDENTITY$(f(),\ g())$

1: Convert $g()$ into canonical form.
2: Check if the coefficients of $g()$ match up with the coefficients of $f()$.

Analysis

Deterministic Approach

Function VERIFY-POLYNOMIAL-IDENTITY$(f(),\ g())$

1: Convert $g()$ into canonical form.
2: Check if the coefficients of $g()$ match up with the coefficients of $f()$.

Analysis

What is the running time?

Deterministic Approach

Function VERIFY-POLYNOMIAL-IDENTITY$(f(),\ g())$

1: Convert $g()$ into canonical form.
2: Check if the coefficients of $g()$ match up with the coefficients of $f()$.

Analysis

What is the running time? $\Theta(n^2)$!

Deterministic Approach

Function VERIFY-POLYNOMIAL-IDENTITY($f(), g()$)

1: Convert $g()$ into canonical form.
2: Check if the coefficients of $g()$ match up with the coefficients of $f()$.

Analysis

What is the running time? $\Theta(n^2)$! Can we do better?

A randomized approach

A randomized approach

Randomized Approach

Function RANDOMIZED-VERIFY-POLYNOMIAL-IDENTITY$(f(),\ g())$

Pick an integer r uniformly from the interval $\{1, 2, \ldots, 2 \cdot n\}$.
Compute $s = f(r)$ and $t = g(r)$.
if $(s = t)$ **then**
    $f()$ and $g()$ are identical.
else
    $f()$ and $g()$ are **not** identical.
end if

A randomized approach

Randomized Approach

Function RANDOMIZED-VERIFY-POLYNOMIAL-IDENTITY$(f(),\ g())$

1: Pick an integer r uniformly from the interval $\{1,2,\ldots,2\cdot n\}$.
2: Compute $s = f(r)$ and $t = g(r)$.
3: **if** $(s = t)$ **then**
4: $f()$ and $g()$ are identical.
5: **else**
6: $f()$ and $g()$ are **not** identical.
7: **end if**

Running Time

A randomized approach

Randomized Approach

Function RANDOMIZED-VERIFY-POLYNOMIAL-IDENTITY$(f(),\ g())$

```
Pick an integer r uniformly from the interval {1,2,...,2·n}.
Compute s = f(r) and t = g(r).
if (s = t) then
    f() and g() are identical.
else
    f() and g() are not identical.
end if
```

Running Time

$O(n)$ for computing t. How much time for computing s?

A randomized approach

Randomized Approach

Function RANDOMIZED-VERIFY-POLYNOMIAL-IDENTITY($f(),\ g()$)

1: Pick an integer r uniformly from the interval $\{1,2,\ldots,2\cdot n\}$.
2: Compute $s = f(r)$ and $t = g(r)$.
3: **if** $(s = t)$ **then**
4: $f()$ and $g()$ are identical.
5: **else**
6: $f()$ and $g()$ are **not** identical.
7: **end if**

Running Time

$O(n)$ for computing t. How much time for computing s? $O(n^2)$?

A randomized approach

Randomized Approach

Function RANDOMIZED-VERIFY-POLYNOMIAL-IDENTITY($f(),\ g()$)

Pick an integer r uniformly from the interval $\{1, 2, \ldots, 2 \cdot n\}$.
Compute $s = f(r)$ and $t = g(r)$.
if $(s = t)$ **then**
    $f()$ and $g()$ are identical.
else
    $f()$ and $g()$ are **not** identical.
end if

Running Time

$O(n)$ for computing t. How much time for computing s? $O(n^2)$? Actually $\Theta(n)$!

Correctness analysis

Correctness

Correctness analysis

Correctness

(i) When $f() = g()$, does the randomized algorithm make a mistake?

Correctness analysis

Correctness

(i) When $f() = g()$, does the randomized algorithm make a mistake?

(ii) When $f() \neq g()$, does the randomized algorithm make a mistake?

Correctness analysis

Correctness

(i) When $f() = g()$, does the randomized algorithm make a mistake?

(ii) When $f() \neq g()$, does the randomized algorithm make a mistake?

(iii) What can you conclude when the randomized algorithm declares that $f() \neq g()$?

Correctness analysis

Correctness

(i) When $f() = g()$, does the randomized algorithm make a mistake?

(ii) When $f() \neq g()$, does the randomized algorithm make a mistake?

(iii) What can you conclude when the randomized algorithm declares that $f() \neq g()$?

(iv) What can you conclude when the randomized algorithm declares that $f() = g()$?

Correctness analysis

Correctness

(i) When $f() = g()$, does the randomized algorithm make a mistake?

(ii) When $f() \neq g()$, does the randomized algorithm make a mistake?

(iii) What can you conclude when the randomized algorithm declares that $f() \neq g()$?

(iv) What can you conclude when the randomized algorithm declares that $f() = g()$?

Note

If the algorithm declares that $f() \neq g()$, then the algorithm is correct.

Correctness analysis

Correctness

(i) When $f() = g()$, does the randomized algorithm make a mistake?
(ii) When $f() \neq g()$, does the randomized algorithm make a mistake?
(iii) What can you conclude when the randomized algorithm declares that $f() \neq g()$?
(iv) What can you conclude when the randomized algorithm declares that $f() = g()$?

Note

If the algorithm declares that $f() \neq g()$, then the algorithm is correct. If the algorithm declares that $f() = g()$, then it is possible that $f() \neq g()$.

Correctness analysis

Correctness

(i) When $f() = g()$, does the randomized algorithm make a mistake?

(ii) When $f() \neq g()$, does the randomized algorithm make a mistake?

(iii) What can you conclude when the randomized algorithm declares that $f() \neq g()$?

(iv) What can you conclude when the randomized algorithm declares that $f() = g()$?

Note

If the algorithm declares that $f() \neq g()$, then the algorithm is correct. If the algorithm declares that $f() = g()$, then it is possible that $f() \neq g()$. We need to bound the probability of this event.

Bounding the error probability

Bounding the error probability

Definition

The values at which a polynomial function evaluates to zero are called its roots.

Bounding the error probability

Definition

The values at which a polynomial function evaluates to zero are called its roots.

Theorem

A polynomial of degree n has exactly n roots

Bounding the error probability

Definition

The values at which a polynomial function evaluates to zero are called its roots.

Theorem

A polynomial of degree n has exactly n roots (not necessarily distinct).

Bounding the error probability

Definition

The values at which a polynomial function evaluates to zero are called its roots.

Theorem

A polynomial of degree n has exactly n roots (not necessarily distinct). (Fundamental theorem of algebra).

Observation

Two distinct polynomials $f()$ and $g()$ can be equal only at the roots of the polynomial $f() - g()$.

Bounding the error probability

Definition

The values at which a polynomial function evaluates to zero are called its roots.

Theorem

A polynomial of degree n has exactly n roots (not necessarily distinct). (Fundamental theorem of algebra).

Observation

Two distinct polynomials $f()$ *and* $g()$ *can be equal only at the roots of the polynomial* $f() - g()$*. The polynomial* $f() - g()$ *has at most n distinct roots.*

Bounding the error probability (contd.)

Bounding the error probability (contd.)

Observation

The only way for the randomized algorithm to give an incorrect answer when $f() \neq g()$, is if the integer r that it picked, is a root of the polynomial $f() - g()$.

Bounding the error probability (contd.)

Observation

The only way for the randomized algorithm to give an incorrect answer when $f() \neq g()$*, is if the integer* r *that it picked, is a root of the polynomial* $f() - g()$*. There can be at most* n *roots of* $f() - g()$ *in the range* $\{1, 2, \ldots, 2 \cdot n\}$.

Bounding the error probability (contd.)

Observation

The only way for the randomized algorithm to give an incorrect answer when $f() \neq g()$*, is if the integer* r *that it picked, is a root of the polynomial* $f() - g()$*. There can be at most* n *roots of* $f() - g()$ *in the range* $\{1, 2, \ldots, 2 \cdot n\}$*. Since* r *is chosen uniformly and at random,*

Bounding the error probability (contd.)

Observation

The only way for the randomized algorithm to give an incorrect answer when $f() \neq g()$, is if the integer r that it picked, is a root of the polynomial $f() - g()$. There can be at most n roots of $f() - g()$ in the range $\{1, 2, \ldots, 2 \cdot n\}$. Since r is chosen uniformly and at random, the probability that r is a root of $f() - g()$ is at most $\frac{n}{2 \cdot n} = \frac{1}{2}$.

Bounding the error probability (contd.)

Observation

The only way for the randomized algorithm to give an incorrect answer when $f() \neq g()$, is if the integer r that it picked, is a root of the polynomial $f() - g()$. There can be at most n roots of $f() - g()$ in the range $\{1, 2, \ldots, 2 \cdot n\}$. Since r is chosen uniformly and at random, the probability that r is a root of $f() - g()$ is at most $\frac{n}{2 \cdot n} = \frac{1}{2}$.

Theorem

On "yes" instances, the randomized algorithm does not err. On "no" instances, the probability that the algorithm errs is at most $\frac{1}{2}$.

Matrix multiplication and Verification

Matrix multiplication and Verification

Problem Statement

Given 2 square $n \times n$ matrices $\mathbf{A}$ and $\mathbf{B}$, compute $\mathbf{C} = \mathbf{A} \cdot \mathbf{B}$.

Matrix multiplication and Verification

Problem Statement

Given 2 square $n \times n$ matrices $\mathbf{A}$ and $\mathbf{B}$, compute $\mathbf{C} = \mathbf{A} \cdot \mathbf{B}$.

Assumption

We will assume that all entries in $\mathbf{A}$ *and* $\mathbf{B}$ *belong to the set* $\{0,1\}$.

Matrix multiplication and Verification

Problem Statement

Given 2 square $n \times n$ matrices **A** and **B**, compute $\mathbf{C} = \mathbf{A} \cdot \mathbf{B}$.

Assumption

We will assume that all entries in **A** *and* **B** *belong to the set* $\{0, 1\}$.

Deterministic Approaches

Matrix multiplication and Verification

Problem Statement

Given 2 square $n \times n$ matrices $\mathbf{A}$ and $\mathbf{B}$, compute $\mathbf{C} = \mathbf{A} \cdot \mathbf{B}$.

Assumption

We will assume that all entries in $\mathbf{A}$ *and* $\mathbf{B}$ *belong to the set* $\{0, 1\}$.

Deterministic Approaches

(i) Naive.

Matrix multiplication and Verification

Problem Statement

Given 2 square $n \times n$ matrices **A** and **B**, compute $\mathbf{C} = \mathbf{A} \cdot \mathbf{B}$.

Assumption

We will assume that all entries in **A** *and* **B** *belong to the set* $\{0, 1\}$.

Deterministic Approaches

(i) Naive.
(ii) Strassen.

Matrix multiplication and Verification

Problem Statement

Given 2 square $n \times n$ matrices **A** and **B**, compute $\mathbf{C} = \mathbf{A} \cdot \mathbf{B}$.

Assumption

We will assume that all entries in **A** *and* **B** *belong to the set* $\{0, 1\}$.

Deterministic Approaches

(i) Naive.
(ii) Strassen.

How to verify Strassen?

A randomized approach to verification

Randomized Approach

Function RANDOMIZED-VERIFY-MATRIX-PRODUCT($\mathbf{C}, \mathbf{A}, \mathbf{B}$)

1: Pick a vector **r** uniformly from the box $\{0, 1\}^n$.
2: Compute $s = \mathbf{A} \cdot \mathbf{B} \cdot \mathbf{r}$ and $t = \mathbf{C} \cdot r$.
3: **if** $(s = t)$ **then**
4:     **C** is the product of **A** and **B**.
5: **else**
6:     **C** is **not** the product of **A** and **B**.
7: **end if**

A randomized approach to verification

Randomized Approach

Function RANDOMIZED-VERIFY-MATRIX-PRODUCT($\mathbf{C}, \mathbf{A}, \mathbf{B}$)

1: Pick a vector **r** uniformly from the box $\{0,1\}^n$.
2: Compute $s = \mathbf{A} \cdot \mathbf{B} \cdot \mathbf{r}$ and $t = \mathbf{C} \cdot r$.
3: **if** $(s = t)$ **then**
4:     **C** is the product of **A** and **B**.
5: **else**
6:     **C** is **not** the product of **A** and **B**.
7: **end if**

Running Time

A randomized approach to verification

Randomized Approach

Function RANDOMIZED-VERIFY-MATRIX-PRODUCT($\mathbf{C}, \mathbf{A}, \mathbf{B}$)

1: Pick a vector **r** uniformly from the box $\{0, 1\}^n$.
2: Compute $s = \mathbf{A} \cdot \mathbf{B} \cdot \mathbf{r}$ and $t = \mathbf{C} \cdot r$.
3: **if** $(s = t)$ **then**
4:     **C** is the product of **A** and **B**.
5: **else**
6:     **C** is **not** the product of **A** and **B**.
7: **end if**

Running Time

$O(n^2)$ for computing t. How much time for computing s?

A randomized approach to verification

Randomized Approach

Function RANDOMIZED-VERIFY-MATRIX-PRODUCT($\mathbf{C}, \mathbf{A}, \mathbf{B}$)

Pick a vector $\mathbf{r}$ uniformly from the box $\{0,1\}^n$.
Compute $s = \mathbf{A} \cdot \mathbf{B} \cdot \mathbf{r}$ and $t = \mathbf{C} \cdot r$.
if $(s = t)$ **then**
    $\mathbf{C}$ is the product of $\mathbf{A}$ and $\mathbf{B}$.
else
    $\mathbf{C}$ is **not** the product of $\mathbf{A}$ and $\mathbf{B}$.
end if

Running Time

$O(n^2)$ for computing t. How much time for computing s? $O(n^3 + n^2 = n^3)$?

A randomized approach to verification

Randomized Approach

Function RANDOMIZED-VERIFY-MATRIX-PRODUCT$(\mathbf{C}, \mathbf{A}, \mathbf{B})$

Pick a vector **r** uniformly from the box $\{0,1\}^n$.
Compute $s = \mathbf{A} \cdot \mathbf{B} \cdot \mathbf{r}$ and $t = \mathbf{C} \cdot r$.
if $(s = t)$ **then**
    C is the product of **A** and **B**.
else
    C is **not** the product of **A** and **B**.
end if

Running Time

$O(n^2)$ for computing t. How much time for computing s? $O(n^3 + n^2 = n^3)$? Actually $O(n^2)$!

Analysis

Analysis

Observation

If $\mathbf{A} \cdot \mathbf{B} = \mathbf{C}$*, then the randomized algorithm does not err.*

Analysis

Observation

If $\mathbf{A} \cdot \mathbf{B} = \mathbf{C}$*, then the randomized algorithm does not err. If* $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{C}$*, then the randomized algorithm could err, since it could be the case that* $\mathbf{A} \cdot \mathbf{B} \cdot \mathbf{r} = \mathbf{C} \cdot \mathbf{r}$*, for the chosen* $\mathbf{r}$*,*

Analysis

Observation

If $\mathbf{A} \cdot \mathbf{B} = \mathbf{C}$*, then the randomized algorithm does not err. If* $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{C}$*, then the randomized algorithm could err, since it could be the case that* $\mathbf{A} \cdot \mathbf{B} \cdot \mathbf{r} = \mathbf{C} \cdot \mathbf{r}$*, for the chosen* $\mathbf{r}$*, but there exists some other vector* $\mathbf{u}$*, such that* $\mathbf{A} \cdot \mathbf{B} \cdot \mathbf{u} \neq \mathbf{C} \cdot \mathbf{u}$*.*

Analysis

Observation

If $\mathbf{A} \cdot \mathbf{B} = \mathbf{C}$, *then the randomized algorithm does not err. If* $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{C}$, *then the randomized algorithm could err, since it could be the case that* $\mathbf{A} \cdot \mathbf{B} \cdot \mathbf{r} = \mathbf{C} \cdot \mathbf{r}$, *for the chosen* $\mathbf{r}$, *but there exists some other vector* $\mathbf{u}$, *such that* $\mathbf{A} \cdot \mathbf{B} \cdot \mathbf{u} \neq \mathbf{C} \cdot \mathbf{u}$. *We need to bound the probability of the error, over the random choices made by the algorithm.*

Analysis

Observation

If $\mathbf{A} \cdot \mathbf{B} = \mathbf{C}$*, then the randomized algorithm does not err. If* $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{C}$*, then the randomized algorithm could err, since it could be the case that* $\mathbf{A} \cdot \mathbf{B} \cdot \mathbf{r} = \mathbf{C} \cdot \mathbf{r}$*, for the chosen* $\mathbf{r}$*, but there exists some other vector* $\mathbf{u}$*, such that* $\mathbf{A} \cdot \mathbf{B} \cdot \mathbf{u} \neq \mathbf{C} \cdot \mathbf{u}$*. We need to bound the probability of the error, over the random choices made by the algorithm.*

Main idea

We will show that if $\mathbf{r}$ is chosen uniformly from $\{0,1\}^n$, then the probability that $\mathbf{A} \cdot \mathbf{B} \cdot \mathbf{r} = \mathbf{C} \cdot \mathbf{r}$, when $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{C}$ is at most $\frac{1}{2}$.

Analysis

Observation

If $\mathbf{A} \cdot \mathbf{B} = \mathbf{C}$, then the randomized algorithm does not err. If $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{C}$, then the randomized algorithm could err, since it could be the case that $\mathbf{A} \cdot \mathbf{B} \cdot \mathbf{r} = \mathbf{C} \cdot \mathbf{r}$, for the chosen $\mathbf{r}$, but there exists some other vector $\mathbf{u}$, such that $\mathbf{A} \cdot \mathbf{B} \cdot \mathbf{u} \neq \mathbf{C} \cdot \mathbf{u}$. We need to bound the probability of the error, over the random choices made by the algorithm.

Main idea

We will show that if $\mathbf{r}$ is chosen uniformly from $\{0,1\}^n$, then the probability that $\mathbf{A} \cdot \mathbf{B} \cdot \mathbf{r} = \mathbf{C} \cdot \mathbf{r}$, when $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{C}$ is at most $\frac{1}{2}$.

Lemma

There is no difference between choosing $\mathbf{r}$ uniformly from $\{0,1\}^n$ and choosing each of its components uniformly over the set $\{0,1\}$.

Analysis (contd.)

Analysis (contd.)

Bounding the error

Analysis (contd.)

Bounding the error

(i) Assume that $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{C}$, but that $\mathbf{A} \cdot \mathbf{B} \cdot \mathbf{r} = \mathbf{C} \cdot \mathbf{r}$.

Analysis (contd.)

Bounding the error

(i) Assume that $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{C}$, but that $\mathbf{A} \cdot \mathbf{B} \cdot \mathbf{r} = \mathbf{C} \cdot \mathbf{r}$.

(ii) Let $\mathbf{D} = \mathbf{A} \cdot \mathbf{B} - \mathbf{C}$.

Analysis (contd.)

Bounding the error

(i) Assume that $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{C}$, but that $\mathbf{A} \cdot \mathbf{B} \cdot \mathbf{r} = \mathbf{C} \cdot \mathbf{r}$.

(ii) Let $\mathbf{D} = \mathbf{A} \cdot \mathbf{B} - \mathbf{C}$. Can $\mathbf{D}$ be $\mathbf{0}$?

Analysis (contd.)

Bounding the error

(i) Assume that $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{C}$, but that $\mathbf{A} \cdot \mathbf{B} \cdot \mathbf{r} = \mathbf{C} \cdot \mathbf{r}$.

(ii) Let $\mathbf{D} = \mathbf{A} \cdot \mathbf{B} - \mathbf{C}$. Can $\mathbf{D}$ be $\mathbf{0}$?

(iii) Without loss of generality assume that the first element of the first row of $\mathbf{D}$, i.e., d_{11} is not 0.

Analysis (contd.)

Bounding the error

(i) Assume that $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{C}$, but that $\mathbf{A} \cdot \mathbf{B} \cdot \mathbf{r} = \mathbf{C} \cdot \mathbf{r}$.

(ii) Let $\mathbf{D} = \mathbf{A} \cdot \mathbf{B} - \mathbf{C}$. Can $\mathbf{D}$ be $\mathbf{0}$?

(iii) Without loss of generality assume that the first element of the first row of $\mathbf{D}$, i.e., d_{11} is not 0.

(iv) Since $\mathbf{D} \cdot \mathbf{r} = 0$, it means that

Analysis (contd.)

Bounding the error

(i) Assume that $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{C}$, but that $\mathbf{A} \cdot \mathbf{B} \cdot \mathbf{r} = \mathbf{C} \cdot \mathbf{r}$.

(ii) Let $\mathbf{D} = \mathbf{A} \cdot \mathbf{B} - \mathbf{C}$. Can $\mathbf{D}$ be $\mathbf{0}$?

(iii) Without loss of generality assume that the first element of the first row of $\mathbf{D}$, i.e., d_{11} is not 0.

(iv) Since $\mathbf{D} \cdot \mathbf{r} = 0$, it means that $\sum_{i=1}^{n} d_{1i} \cdot r_i = 0$.

Analysis (contd.)

Bounding the error

(i) Assume that $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{C}$, but that $\mathbf{A} \cdot \mathbf{B} \cdot \mathbf{r} = \mathbf{C} \cdot \mathbf{r}$.

(ii) Let $\mathbf{D} = \mathbf{A} \cdot \mathbf{B} - \mathbf{C}$. Can $\mathbf{D}$ be $\mathbf{0}$?

(iii) Without loss of generality assume that the first element of the first row of $\mathbf{D}$, i.e., d_{11} is not 0.

(iv) Since $\mathbf{D} \cdot \mathbf{r} = 0$, it means that $\sum_{i=1}^{n} d_{1i} \cdot r_i = 0$.

(v) Therefore, $r_1 = -\frac{\sum_{j=2}^{n} d_{ij} \cdot r_j}{d_{11}}$.

Analysis (contd.)

Bounding the error

(i) Assume that $\mathbf{A}\cdot\mathbf{B}\neq\mathbf{C}$, but that $\mathbf{A}\cdot\mathbf{B}\cdot\mathbf{r}=\mathbf{C}\cdot\mathbf{r}$.

(ii) Let $\mathbf{D}=\mathbf{A}\cdot\mathbf{B}-\mathbf{C}$. Can $\mathbf{D}$ be $\mathbf{0}$?

(iii) Without loss of generality assume that the first element of the first row of $\mathbf{D}$, i.e., d_{11} is not 0.

(iv) Since $\mathbf{D}\cdot\mathbf{r}=0$, it means that $\sum_{i=1}^{n} d_{1i}\cdot r_i=0$.

(v) Therefore, $r_1=-\frac{\sum_{j=2}^{n} d_{ij}\cdot r_j}{d_{11}}$.

(vi) Assume that $r_2,r_3,\ldots,r_n$ are chosen before r_1.

Analysis (contd.)

Bounding the error

(i) Assume that $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{C}$, but that $\mathbf{A} \cdot \mathbf{B} \cdot \mathbf{r} = \mathbf{C} \cdot \mathbf{r}$.

(ii) Let $\mathbf{D} = \mathbf{A} \cdot \mathbf{B} - \mathbf{C}$. Can $\mathbf{D}$ be $\mathbf{0}$?

(iii) Without loss of generality assume that the first element of the first row of $\mathbf{D}$, i.e., d_{11} is not 0.

(iv) Since $\mathbf{D} \cdot \mathbf{r} = 0$, it means that $\sum_{i=1}^{n} d_{1i} \cdot r_i = 0$.

(v) Therefore, $r_1 = -\frac{\sum_{j=2}^{n} d_{ij} \cdot r_j}{d_{11}}$.

(vi) Assume that $r_2, r_3, \ldots, r_n$ are chosen before r_1. Thus the RHS is fixed and evaluated before r_1 is chosen.

Analysis (contd.)

Bounding the error

(i) Assume that $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{C}$, but that $\mathbf{A} \cdot \mathbf{B} \cdot \mathbf{r} = \mathbf{C} \cdot \mathbf{r}$.

(ii) Let $\mathbf{D} = \mathbf{A} \cdot \mathbf{B} - \mathbf{C}$. Can $\mathbf{D}$ be $\mathbf{0}$?

(iii) Without loss of generality assume that the first element of the first row of $\mathbf{D}$, i.e., d_{11} is not 0.

(iv) Since $\mathbf{D} \cdot \mathbf{r} = 0$, it means that $\sum_{i=1}^{n} d_{1i} \cdot r_i = 0$.

(v) Therefore, $r_1 = -\frac{\sum_{j=2}^{n} d_{ij} \cdot r_j}{d_{11}}$.

(vi) Assume that $r_2, r_3, \ldots, r_n$ are chosen before r_1. Thus the RHS is fixed and evaluated before r_1 is chosen. Thus the probability that the value of r_1 chosen will be equal to the RHS value is at most $\frac{1}{2}$.

Analysis (contd.)

Bounding the error

(i) Assume that $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{C}$, but that $\mathbf{A} \cdot \mathbf{B} \cdot \mathbf{r} = \mathbf{C} \cdot \mathbf{r}$.

(ii) Let $\mathbf{D} = \mathbf{A} \cdot \mathbf{B} - \mathbf{C}$. Can $\mathbf{D}$ be $\mathbf{0}$?

(iii) Without loss of generality assume that the first element of the first row of $\mathbf{D}$, i.e., d_{11} is not 0.

(iv) Since $\mathbf{D} \cdot \mathbf{r} = 0$, it means that $\sum_{i=1}^{n} d_{1i} \cdot r_i = 0$.

(v) Therefore, $r_1 = -\frac{\sum_{j=2}^{n} d_{ij} \cdot r_j}{d_{11}}$.

(vi) Assume that $r_2, r_3, \ldots, r_n$ are chosen before r_1. Thus the RHS is fixed and evaluated before r_1 is chosen. Thus the probability that the value of r_1 chosen will be equal to the RHS value is at most $\frac{1}{2}$.

Note

The above method of analysis is called the Principle of Deferred Decisons.

The Min-cut problem on undirected unweighted graphs

The Min-cut problem on undirected unweighted graphs

Definition

Given an undirected graph $\mathbf{G} = \langle \mathbf{V}, \mathbf{E} \rangle$,

The Min-cut problem on undirected unweighted graphs

Definition

Given an undirected graph $\mathbf{G} = \langle \mathbf{V}, \mathbf{E} \rangle$, a *cut-set* or *cut* is defined as some set of edges $\mathbf{E}' \subseteq \mathbf{E}$,

The Min-cut problem on undirected unweighted graphs

Definition

Given an undirected graph $\mathbf{G} = \langle \mathbf{V}, \mathbf{E} \rangle$, a *cut-set* or *cut* is defined as some set of edges $\mathbf{E}' \subseteq \mathbf{E}$, whose removal disconnects $\mathbf{G}$.

The Min-cut problem on undirected unweighted graphs

Definition

Given an undirected graph $\mathbf{G} = \langle \mathbf{V}, \mathbf{E} \rangle$, a *cut-set* or *cut* is defined as some set of edges $\mathbf{E}' \subseteq \mathbf{E}$, whose removal disconnects $\mathbf{G}$. In other words, $\mathbf{G} = \langle \mathbf{V}, \mathbf{E} - \mathbf{E}' \rangle$ has at least 2 components.

The Min-cut problem on undirected unweighted graphs

Definition

Given an undirected graph $\mathbf{G} = \langle \mathbf{V}, \mathbf{E} \rangle$, a *cut-set* or *cut* is defined as some set of edges $\mathbf{E}' \subseteq \mathbf{E}$, whose removal disconnects $\mathbf{G}$. In other words, $\mathbf{G} = \langle \mathbf{V}, \mathbf{E} - \mathbf{E}' \rangle$ has at least 2 components.

Problem Statement

Given an undirected graph $\mathbf{G} = \langle \mathbf{V}, \mathbf{E} \rangle$, find a cut $\mathbf{E}' \subseteq \mathbf{E}$, such that,

The Min-cut problem on undirected unweighted graphs

Definition

Given an undirected graph $\mathbf{G} = \langle \mathbf{V}, \mathbf{E} \rangle$, a *cut-set* or *cut* is defined as some set of edges $\mathbf{E}' \subseteq \mathbf{E}$, whose removal disconnects $\mathbf{G}$. In other words, $\mathbf{G} = \langle \mathbf{V}, \mathbf{E} - \mathbf{E}' \rangle$ has at least 2 components.

Problem Statement

Given an undirected graph $\mathbf{G} = \langle \mathbf{V}, \mathbf{E} \rangle$, find a cut $\mathbf{E}' \subseteq \mathbf{E}$, such that, $|\mathbf{E}'| \leq |\mathbf{E}''|$, over all cuts $\mathbf{E}'' \subseteq \mathbf{E}$.

The Min-cut problem on undirected unweighted graphs

Definition

Given an undirected graph $\mathbf{G} = \langle \mathbf{V}, \mathbf{E} \rangle$, a *cut-set* or *cut* is defined as some set of edges $\mathbf{E}' \subseteq \mathbf{E}$, whose removal disconnects $\mathbf{G}$. In other words, $\mathbf{G} = \langle \mathbf{V}, \mathbf{E} - \mathbf{E}' \rangle$ has at least 2 components.

Problem Statement

Given an undirected graph $\mathbf{G} = \langle \mathbf{V}, \mathbf{E} \rangle$, find a cut $\mathbf{E}' \subseteq \mathbf{E}$, such that, $|\mathbf{E}'| \leq |\mathbf{E}''|$, over all cuts $\mathbf{E}'' \subseteq \mathbf{E}$.

Max-Flow approach

The Min-cut problem on undirected unweighted graphs

Definition

Given an undirected graph $\mathbf{G} = \langle \mathbf{V}, \mathbf{E} \rangle$, a *cut-set* or *cut* is defined as some set of edges $\mathbf{E}' \subseteq \mathbf{E}$, whose removal disconnects $\mathbf{G}$. In other words, $\mathbf{G} = \langle \mathbf{V}, \mathbf{E} - \mathbf{E}' \rangle$ has at least 2 components.

Problem Statement

Given an undirected graph $\mathbf{G} = \langle \mathbf{V}, \mathbf{E} \rangle$, find a cut $\mathbf{E}' \subseteq \mathbf{E}$, such that, $|\mathbf{E}'| \leq |\mathbf{E}''|$, over all cuts $\mathbf{E}'' \subseteq \mathbf{E}$.

Max-Flow approach

(i) Difference between $s - t$ Min-cut and Global Min-cut.

The Min-cut problem on undirected unweighted graphs

Definition

Given an undirected graph $\mathbf{G} = \langle \mathbf{V}, \mathbf{E} \rangle$, a *cut-set* or *cut* is defined as some set of edges $\mathbf{E}' \subseteq \mathbf{E}$, whose removal disconnects $\mathbf{G}$. In other words, $\mathbf{G} = \langle \mathbf{V}, \mathbf{E} - \mathbf{E}' \rangle$ has at least 2 components.

Problem Statement

Given an undirected graph $\mathbf{G} = \langle \mathbf{V}, \mathbf{E} \rangle$, find a cut $\mathbf{E}' \subseteq \mathbf{E}$, such that, $|\mathbf{E}'| \leq |\mathbf{E}''|$, over all cuts $\mathbf{E}'' \subseteq \mathbf{E}$.

Max-Flow approach

(i) Difference between $s - t$ Min-cut and Global Min-cut.

(ii) Solving $s - t$ Min-cut.

The Min-cut problem on undirected unweighted graphs

Definition

Given an undirected graph $\mathbf{G} = \langle \mathbf{V}, \mathbf{E} \rangle$, a *cut-set* or *cut* is defined as some set of edges $\mathbf{E}' \subseteq \mathbf{E}$, whose removal disconnects $\mathbf{G}$. In other words, $\mathbf{G} = \langle \mathbf{V}, \mathbf{E} - \mathbf{E}' \rangle$ has at least 2 components.

Problem Statement

Given an undirected graph $\mathbf{G} = \langle \mathbf{V}, \mathbf{E} \rangle$, find a cut $\mathbf{E}' \subseteq \mathbf{E}$, such that, $|\mathbf{E}'| \leq |\mathbf{E}''|$, over all cuts $\mathbf{E}'' \subseteq \mathbf{E}$.

Max-Flow approach

(i) Difference between $s - t$ Min-cut and Global Min-cut.
(ii) Solving $s - t$ Min-cut.
(iii) Reducing Global Min-cut to $s - t$ Min-cut.

The Randomized approach

The Randomized approach

The Contract Operation

Function CONTRACT-EDGE(**G**, *e*)

1: {We will contract edge *e* in **G**.}
2: Let *u* and *v* denote the end-points of *e*.
3: Identify *u* and *v* into a single vertex.
4: Remove all self-loops.
5: Keep all parallel edges.

The Randomized approach

The Contract Operation

Function CONTRACT-EDGE(**G**, e)

1: {We will contract edge e in **G**.}
2: Let u and v denote the end-points of e.
3: Identify u and v into a single vertex.
4: Remove all self-loops.
5: Keep all parallel edges.

Note

CONTRACT-EDGE() *reduces the number of vertices in the graph by* 1.

The Randomized approach

The Contract Operation

Function CONTRACT-EDGE(**G**, e)

1: {We will contract edge e in **G**.}
2: Let u and v denote the end-points of e.
3: Identify u and v into a single vertex.
4: Remove all self-loops.
5: Keep all parallel edges.

Note

CONTRACT-EDGE() *reduces the number of vertices in the graph by* 1*. So we can call* CONTRACT-EDGE() *at most* $(n-2)$ *times, before a cut of the graph is revealed, i.e.,*

The Randomized approach

The Contract Operation

Function CONTRACT-EDGE(**G**, e)

1: {We will contract edge e in **G**.}
2: Let u and v denote the end-points of e.
3: Identify u and v into a single vertex.
4: Remove all self-loops.
5: Keep all parallel edges.

Note

CONTRACT-EDGE() *reduces the number of vertices in the graph by* 1*. So we can call* CONTRACT-EDGE() *at most* $(n-2)$ *times, before a cut of the graph is revealed, i.e., all the edges are between two giant vertices.*

Randomized Min-cut

Randomized Min-cut

The Randomized Algorithm

Function RANDOMIZED MIN-CUT($\mathbf{G} = \langle \mathbf{V}, \mathbf{E} \rangle$)

while there are more than 2 vertices in **G do**
    Pick an edge e uniformly and at random from the edge set **E**
    CONTRACT-EDGE($\mathbf{G}, e$)
end while
Let a and b denote the last two vertices that remain.
Output the edges between a and b as the min-cut of **G**.

Analysis

Analysis

Focus

Assume that **G** has a min-cut of size k.

Analysis

Focus

Assume that **G** has a min-cut of size k. Note that there could be several min-cuts with size k.

Analysis

Focus

Assume that **G** has a min-cut of size k. Note that there could be several min-cuts with size k. Without loss of generality, we focus on a specific cut, say C.

Analysis

Focus

Assume that **G** has a min-cut of size k. Note that there could be several min-cuts with size k. Without loss of generality, we focus on a specific cut, say C. The only way that the randomized algorithm produces the correct answer is if no edge of C was contracted during the $(n-2)$ iterations.

Analysis

Focus

Assume that **G** has a min-cut of size k. Note that there could be several min-cuts with size k. Without loss of generality, we focus on a specific cut, say C. The only way that the randomized algorithm produces the correct answer is if no edge of C was contracted during the $(n-2)$ iterations. The question of interest then is:

Analysis

Focus

Assume that **G** has a min-cut of size k. Note that there could be several min-cuts with size k. Without loss of generality, we focus on a specific cut, say C. The only way that the randomized algorithm produces the correct answer is if no edge of C was contracted during the $(n-2)$ iterations. The question of interest then is: What is the probability that C survived the $(n-2)$ edge contractions of the randomized algorithm?

Analysis

Focus

Assume that **G** has a min-cut of size k. Note that there could be several min-cuts with size k. Without loss of generality, we focus on a specific cut, say C. The only way that the randomized algorithm produces the correct answer is if no edge of C was contracted during the $(n-2)$ iterations. The question of interest then is: What is the probability that C survived the $(n-2)$ edge contractions of the randomized algorithm?

Analytic observations

Analysis

Focus

Assume that **G** has a min-cut of size k. Note that there could be several min-cuts with size k. Without loss of generality, we focus on a specific cut, say C. The only way that the randomized algorithm produces the correct answer is if no edge of C was contracted during the $(n-2)$ iterations. The question of interest then is: What is the probability that C survived the $(n-2)$ edge contractions of the randomized algorithm?

Analytic observations

Let $\mathbf{E_i}$ denote the event that no edge of C was contracted in iteration i.

Analysis

Focus

Assume that **G** has a min-cut of size k. Note that there could be several min-cuts with size k. Without loss of generality, we focus on a specific cut, say C. The only way that the randomized algorithm produces the correct answer is if no edge of C was contracted during the $(n-2)$ iterations. The question of interest then is: What is the probability that C survived the $(n-2)$ edge contractions of the randomized algorithm?

Analytic observations

Let $\mathbf{E_i}$ denote the event that no edge of C was contracted in iteration i. Clearly, the event that we are interested in is

Analysis

Focus

Assume that **G** has a min-cut of size k. Note that there could be several min-cuts with size k. Without loss of generality, we focus on a specific cut, say C. The only way that the randomized algorithm produces the correct answer is if no edge of C was contracted during the $(n-2)$ iterations. The question of interest then is: What is the probability that C survived the $(n-2)$ edge contractions of the randomized algorithm?

Analytic observations

Let $\mathbf{E_i}$ denote the event that no edge of C was contracted in iteration i. Clearly, the event that we are interested in is $\cap_{i=1}^{n-2} E_i$ with associated probability $P(\cap_{i=1}^{n-2} E_i)$.

Analysis

Focus

Assume that **G** has a min-cut of size k. Note that there could be several min-cuts with size k. Without loss of generality, we focus on a specific cut, say C. The only way that the randomized algorithm produces the correct answer is if no edge of C was contracted during the $(n-2)$ iterations. The question of interest then is: What is the probability that C survived the $(n-2)$ edge contractions of the randomized algorithm?

Analytic observations

Let $\mathbf{E_i}$ denote the event that no edge of C was contracted in iteration i. Clearly, the event that we are interested in is $\cap_{i=1}^{n-2} E_i$ with associated probability $P(\cap_{i=1}^{n-2} E_i)$. How do we compute this probability?

Analysis

Focus

Assume that **G** has a min-cut of size k. Note that there could be several min-cuts with size k. Without loss of generality, we focus on a specific cut, say C. The only way that the randomized algorithm produces the correct answer is if no edge of C was contracted during the $(n-2)$ iterations. The question of interest then is: What is the probability that C survived the $(n-2)$ edge contractions of the randomized algorithm?

Analytic observations

Let $\mathbf{E_i}$ denote the event that no edge of C was contracted in iteration i. Clearly, the event that we are interested in is $\cap_{i=1}^{n-2} E_i$ with associated probability $P(\cap_{i=1}^{n-2} E_i)$. How do we compute this probability? By noting that
$P(\cap_{i=1}^{n-2} E_i) = P(E_1) \cdot P(E_2 \mid E_1) \cdot P(E_3 \mid E_1 \cap E_2) \ldots \cdot P(E_n \mid \cap_{i=1}^{n-1} E_i)$!

Analysis (contd.)

Analytic Observations

Analysis (contd.)

Analytic Observations

(i) What is the minimum degree of a vertex in **G**?

Analysis (contd.)

Analytic Observations

(i) What is the minimum degree of a vertex in **G**? $\geq k$.

Analysis (contd.)

Analytic Observations

(i) What is the minimum degree of a vertex in **G**? $\geq k$.

(ii) What is $P(E_1)$?

Analysis (contd.)

Analytic Observations

(i) What is the minimum degree of a vertex in **G**? $\geq k$.

(ii) What is $P(E_1)$? $\geq (1 - \frac{2}{n})$.

Analysis (contd.)

Analytic Observations

(i) What is the minimum degree of a vertex in **G**? $\geq k$.

(ii) What is $P(E_1)$? $\geq (1 - \frac{2}{n})$.

(iii) What is $P(E_2 \mid E_1)$?

Analysis (contd.)

Analytic Observations

(i) What is the minimum degree of a vertex in **G**? $\geq k$.

(ii) What is $P(E_1)$? $\geq (1 - \frac{2}{n})$.

(iii) What is $P(E_2 \mid E_1)$? $\geq (1 - \frac{2}{n-1})$.

Analysis (contd.)

Analytic Observations

(i) What is the minimum degree of a vertex in **G**? $\geq k$.

(ii) What is $P(E_1)$? $\geq (1 - \frac{2}{n})$.

(iii) What is $P(E_2 \,|\, E_1)$? $\geq (1 - \frac{2}{n-1})$.

(iv) What is $P(E_i \,|\, \cap_{j=1}^{i-1} E_j)$?

Analysis (contd.)

Analytic Observations

(i) What is the minimum degree of a vertex in **G**? $\geq k$.

(ii) What is $P(E_1)$? $\geq (1 - \frac{2}{n})$.

(iii) What is $P(E_2 \mid E_1)$? $\geq (1 - \frac{2}{n-1})$.

(iv) What is $P(E_i \mid \cap_{j=1}^{i-1} E_j)$? $(1 - \frac{2}{n-i+1})$.

Analysis (contd.)

Analytic Observations

(i) What is the minimum degree of a vertex in **G**? $\geq k$.

(ii) What is $P(E_1)$? $\geq (1-\frac{2}{n})$.

(iii) What is $P(E_2 \mid E_1)$? $\geq (1-\frac{2}{n-1})$.

(iv) What is $P(E_i \mid \cap_{j=1}^{i-1} E_j)$? $(1-\frac{2}{n-i+1})$.

(v) What is $P(\cap_{i=1}^{n-2} E_i)$?

Analysis (contd.)

Analytic Observations

(i) What is the minimum degree of a vertex in **G**? $\geq k$.

(ii) What is $P(E_1)$? $\geq (1 - \frac{2}{n})$.

(iii) What is $P(E_2 \mid E_1)$? $\geq (1 - \frac{2}{n-1})$.

(iv) What is $P(E_i \mid \cap_{j=1}^{i-1} E_j)$? $(1 - \frac{2}{n-i+1})$.

(v) What is $P(\cap_{i=1}^{n-2} E_i)$? $\Pi_{i=1}^{n-2}(1 - \frac{2}{n-i+1})$

Analysis (contd.)

Analytic Observations

(i) What is the minimum degree of a vertex in **G**? $\geq k$.

(ii) What is $P(E_1)$? $\geq (1 - \frac{2}{n})$.

(iii) What is $P(E_2 \mid E_1)$? $\geq (1 - \frac{2}{n-1})$.

(iv) What is $P(E_i \mid \cap_{j=1}^{i-1} E_j)$? $(1 - \frac{2}{n-i+1})$.

(v) What is $P(\cap_{i=1}^{n-2} E_i)$? $\Pi_{i=1}^{n-2}(1 - \frac{2}{n-i+1}) \geq \frac{2}{n \cdot (n-1)}$.

Analysis (contd.)

Analytic Observations

(i) What is the minimum degree of a vertex in **G**? $\geq k$.

(ii) What is $P(E_1)$? $\geq (1 - \frac{2}{n})$.

(iii) What is $P(E_2 \mid E_1)$? $\geq (1 - \frac{2}{n-1})$.

(iv) What is $P(E_i \mid \cap_{j=1}^{i-1} E_j)$? $(1 - \frac{2}{n-i+1})$.

(v) What is $P(\cap_{i=1}^{n-2} E_i)$? $\Pi_{i=1}^{n-2}(1 - \frac{2}{n-i+1}) \geq \frac{2}{n \cdot (n-1)}$.

Note

The probability that the randomized algorithm produces the min-cut of the graph is at least $\frac{2}{n \cdot (n-1)}$.

Analysis (contd.)

Analytic Observations

(i) What is the minimum degree of a vertex in **G**? $\geq k$.

(ii) What is $P(E_1)$? $\geq (1-\frac{2}{n})$.

(iii) What is $P(E_2 \mid E_1)$? $\geq (1-\frac{2}{n-1})$.

(iv) What is $P(E_i \mid \cap_{j=1}^{i-1} E_j)$? $(1-\frac{2}{n-i+1})$.

(v) What is $P(\cap_{i=1}^{n-2} E_i)$? $\Pi_{i=1}^{n-2}(1-\frac{2}{n-i+1}) \geq \frac{2}{n \cdot (n-1)}$.

Note

The probability that the randomized algorithm produces the min-cut of the graph is at least $\frac{2}{n \cdot (n-1)}$. *Hence, the probability that the algorithm does not produce the min-cut of the graph in* $n \cdot (n-1) \cdot \ln n$ *independent runs is at most:*

Analysis (contd.)

Analytic Observations

(i) What is the minimum degree of a vertex in **G**? $\geq k$.

(ii) What is $P(E_1)$? $\geq (1 - \frac{2}{n})$.

(iii) What is $P(E_2 \mid E_1)$? $\geq (1 - \frac{2}{n-1})$.

(iv) What is $P(E_i \mid \cap_{j=1}^{i-1} E_j)$? $(1 - \frac{2}{n-i+1})$.

(v) What is $P(\cap_{i=1}^{n-2} E_i)$? $\Pi_{i=1}^{n-2}(1 - \frac{2}{n-i+1}) \geq \frac{2}{n\cdot(n-1)}$.

Note

The probability that the randomized algorithm produces the min-cut of the graph is at least $\frac{2}{n\cdot(n-1)}$*. Hence, the probability that the algorithm does not produce the min-cut of the graph in* $n \cdot (n-1) \cdot \ln n$ *independent runs is at most:*

$$(1 - \frac{2}{n \cdot (n-1)})^{n\cdot(n-1)\cdot\ln n} \quad \leq \quad e^{-2\cdot\ln n}$$

Analysis (contd.)

Analytic Observations

(i) What is the minimum degree of a vertex in **G**? $\geq k$.

(ii) What is $P(E_1)$? $\geq (1 - \frac{2}{n})$.

(iii) What is $P(E_2 \mid E_1)$? $\geq (1 - \frac{2}{n-1})$.

(iv) What is $P(E_i \mid \cap_{j=1}^{i-1} E_j)$? $(1 - \frac{2}{n-i+1})$.

(v) What is $P(\cap_{i=1}^{n-2} E_i)$? $\Pi_{i=1}^{n-2}(1 - \frac{2}{n-i+1}) \geq \frac{2}{n \cdot (n-1)}$.

Note

The probability that the randomized algorithm produces the min-cut of the graph is at least $\frac{2}{n \cdot (n-1)}$*. Hence, the probability that the algorithm does not produce the min-cut of the graph in* $n \cdot (n-1) \cdot \ln n$ *independent runs is at most:*

$$(1 - \frac{2}{n \cdot (n-1)})^{n \cdot (n-1) \cdot \ln n} \quad \leq \quad e^{-2 \cdot \ln n} \ (\text{since } (1-x) \leq e^{-x})$$

Analysis (contd.)

Analytic Observations

(i) What is the minimum degree of a vertex in **G**? $\geq k$.

(ii) What is $P(E_1)$? $\geq (1 - \frac{2}{n})$.

(iii) What is $P(E_2 \mid E_1)$? $\geq (1 - \frac{2}{n-1})$.

(iv) What is $P(E_i \mid \cap_{j=1}^{i-1} E_j)$? $(1 - \frac{2}{n-i+1})$.

(v) What is $P(\cap_{i=1}^{n-2} E_i)$? $\Pi_{i=1}^{n-2}(1 - \frac{2}{n-i+1}) \geq \frac{2}{n\cdot(n-1)}$.

Note

The probability that the randomized algorithm produces the min-cut of the graph is at least $\frac{2}{n\cdot(n-1)}$. *Hence, the probability that the algorithm does not produce the min-cut of the graph in* $n \cdot (n-1) \cdot \ln n$ *independent runs is at most:*

$$
\begin{aligned}
(1 - \frac{2}{n\cdot(n-1)})^{n\cdot(n-1)\cdot\ln n} &\leq e^{-2\cdot\ln n} \quad (\text{since } (1-x) \leq e^{-x}) \\
&= \frac{1}{n^2}
\end{aligned}
$$

Coupon Collector problems

Coupon Collector problems

Problem Statements

Coupon Collector problems

Problem Statements

You are required to collect coupons in a series of iterations.

Coupon Collector problems

Problem Statements

You are required to collect coupons in a series of iterations. Assume that each coupon belongs to one of N types, where N is a fixed number.

Coupon Collector problems

Problem Statements

You are required to collect coupons in a series of iterations. Assume that each coupon belongs to one of N types, where N is a fixed number. The coupons are drawn uniformly and at random from the N coupon types.

Coupon Collector problems

Problem Statements

You are required to collect coupons in a series of iterations. Assume that each coupon belongs to one of N types, where N is a fixed number. The coupons are drawn uniformly and at random from the N coupon types. Several questions are of interest:

Coupon Collector problems

Problem Statements

You are required to collect coupons in a series of iterations. Assume that each coupon belongs to one of N types, where N is a fixed number. The coupons are drawn uniformly and at random from the N coupon types. Several questions are of interest:

(i) For a fixed $t \geq N$, what is the probability that at least t coupons need to be collected to ensure that we have at least one coupon of each type?

Coupon Collector problems

Problem Statements

You are required to collect coupons in a series of iterations. Assume that each coupon belongs to one of N types, where N is a fixed number. The coupons are drawn uniformly and at random from the N coupon types. Several questions are of interest:

(i) For a fixed $t \geq N$, what is the probability that at least t coupons need to be collected to ensure that we have at least one coupon of each type?

(ii) What is the expected number of coupons to be collected, to ensure that each coupon type has been collected?

Coupon Collector problems

Problem Statements

You are required to collect coupons in a series of iterations. Assume that each coupon belongs to one of N types, where N is a fixed number. The coupons are drawn uniformly and at random from the N coupon types. Several questions are of interest:

(i) For a fixed $t \geq N$, what is the probability that at least t coupons need to be collected to ensure that we have at least one coupon of each type?

(ii) What is the expected number of coupons to be collected, to ensure that each coupon type has been collected?

(iii) Suppose you are given a K coupons (usually less than N). What is the expected number of distinct coupon types in these K coupons?

Analysis

Analyzing one coupon collection problem

Analysis

Analyzing one coupon collection problem

We focus on the following question:

Analysis

Analyzing one coupon collection problem

We focus on the following question: What is the expected number of coupons to be collected, to ensure that each coupon type has been collected?

Analysis

Analyzing one coupon collection problem

We focus on the following question: What is the expected number of coupons to be collected, to ensure that each coupon type has been collected? Let X denote the total number of coupons to be drawn, before we have at least one coupon of each of the N distinct types.

Analysis

Analyzing one coupon collection problem

We focus on the following question: What is the expected number of coupons to be collected, to ensure that each coupon type has been collected? Let X denote the total number of coupons to be drawn, before we have at least one coupon of each of the N distinct types. Let X_i denote the number of coupons that need to be drawn, after $(i-1)$ distinct coupon types have been collected, in order to draw a coupon of a type that has not been collected.

Analysis

Analyzing one coupon collection problem

We focus on the following question: What is the expected number of coupons to be collected, to ensure that each coupon type has been collected? Let X denote the total number of coupons to be drawn, before we have at least one coupon of each of the N distinct types. Let X_i denote the number of coupons that need to be drawn, after $(i-1)$ distinct coupon types have been collected, in order to draw a coupon of a type that has not been collected. If p_i denotes the probability of drawing a new coupon after $(i-1)$ coupons have been collected, then clearly, $p_i =$

Analysis

Analyzing one coupon collection problem

We focus on the following question: What is the expected number of coupons to be collected, to ensure that each coupon type has been collected? Let X denote the total number of coupons to be drawn, before we have at least one coupon of each of the N distinct types. Let X_i denote the number of coupons that need to be drawn, after $(i-1)$ distinct coupon types have been collected, in order to draw a coupon of a type that has not been collected. If p_i denotes the probability of drawing a new coupon after $(i-1)$ coupons have been collected, then clearly, $p_i = (1 - \frac{i-1}{N}) =$

Analysis

Analyzing one coupon collection problem

We focus on the following question: What is the expected number of coupons to be collected, to ensure that each coupon type has been collected? Let X denote the total number of coupons to be drawn, before we have at least one coupon of each of the N distinct types. Let X_i denote the number of coupons that need to be drawn, after $(i-1)$ distinct coupon types have been collected, in order to draw a coupon of a type that has not been collected. If p_i denotes the probability of drawing a new coupon after $(i-1)$ coupons have been collected, then clearly, $p_i = (1 - \frac{i-1}{N}) = \frac{N-i+1}{N}$.

Analysis

Analyzing one coupon collection problem

We focus on the following question: What is the expected number of coupons to be collected, to ensure that each coupon type has been collected? Let X denote the total number of coupons to be drawn, before we have at least one coupon of each of the N distinct types. Let X_i denote the number of coupons that need to be drawn, after $(i-1)$ distinct coupon types have been collected, in order to draw a coupon of a type that has not been collected. If p_i denotes the probability of drawing a new coupon after $(i-1)$ coupons have been collected, then clearly, $p_i = (1 - \frac{i-1}{N}) = \frac{N-i+1}{N}$.

Now $X = \sum_{i=1}^{N} X_i$ and hence $E[X] = \sum_{i=1}^{N} E[X_i]$.

Analysis

Analyzing one coupon collection problem

We focus on the following question: What is the expected number of coupons to be collected, to ensure that each coupon type has been collected? Let X denote the total number of coupons to be drawn, before we have at least one coupon of each of the N distinct types. Let X_i denote the number of coupons that need to be drawn, after $(i-1)$ distinct coupon types have been collected, in order to draw a coupon of a type that has not been collected. If p_i denotes the probability of drawing a new coupon after $(i-1)$ coupons have been collected, then clearly, $p_i = (1 - \frac{i-1}{N}) = \frac{N-i+1}{N}$.

Now $X = \sum_{i=1}^{N} X_i$ and hence $E[X] = \sum_{i=1}^{N} E[X_i]$. Observe that each X_i is a geometric random variable with parameter p_i.

Analysis

Analyzing one coupon collection problem

We focus on the following question: What is the expected number of coupons to be collected, to ensure that each coupon type has been collected? Let X denote the total number of coupons to be drawn, before we have at least one coupon of each of the N distinct types. Let X_i denote the number of coupons that need to be drawn, after $(i-1)$ distinct coupon types have been collected, in order to draw a coupon of a type that has not been collected. If p_i denotes the probability of drawing a new coupon after $(i-1)$ coupons have been collected, then clearly, $p_i = (1 - \frac{i-1}{N}) = \frac{N-i+1}{N}$.
Now $X = \sum_{i=1}^{N} X_i$ and hence $E[X] = \sum_{i=1}^{N} E[X_i]$. Observe that each X_i is a geometric random variable with parameter p_i. Therefore,

Analysis

Analyzing one coupon collection problem

We focus on the following question: What is the expected number of coupons to be collected, to ensure that each coupon type has been collected? Let X denote the total number of coupons to be drawn, before we have at least one coupon of each of the N distinct types. Let X_i denote the number of coupons that need to be drawn, after $(i-1)$ distinct coupon types have been collected, in order to draw a coupon of a type that has not been collected. If p_i denotes the probability of drawing a new coupon after $(i-1)$ coupons have been collected, then clearly, $p_i = (1 - \frac{i-1}{N}) = \frac{N-i+1}{N}$.

Now $X = \sum_{i=1}^{N} X_i$ and hence $E[X] = \sum_{i=1}^{N} E[X_i]$. Observe that each X_i is a geometric random variable with parameter p_i. Therefore,

$$E[X] \quad = \quad \sum_{i=1}^{N} E[X_i]$$

Analysis

Analyzing one coupon collection problem

We focus on the following question: What is the expected number of coupons to be collected, to ensure that each coupon type has been collected? Let X denote the total number of coupons to be drawn, before we have at least one coupon of each of the N distinct types. Let X_i denote the number of coupons that need to be drawn, after $(i-1)$ distinct coupon types have been collected, in order to draw a coupon of a type that has not been collected. If p_i denotes the probability of drawing a new coupon after $(i-1)$ coupons have been collected, then clearly, $p_i = (1 - \frac{i-1}{N}) = \frac{N-i+1}{N}$.

Now $X = \sum_{i=1}^{N} X_i$ and hence $E[X] = \sum_{i=1}^{N} E[X_i]$. Observe that each X_i is a geometric random variable with parameter p_i. Therefore,

$$\begin{aligned} E[X] &= \sum_{i=1}^{N} E[X_i] \\ &= \sum_{i=1}^{N} \frac{1}{p_i} \end{aligned}$$

Analysis (contd.)

Analyzing one coupon collection problem (contd.)

Analysis (contd.)

Analyzing one coupon collection problem (contd.)

$$= \sum_{i=1}^{N} \frac{N}{N-i+1}$$

Analysis (contd.)

Analyzing one coupon collection problem (contd.)

$$
\begin{aligned}
&= \sum_{i=1}^{N} \frac{N}{N-i+1} \\
&= N \cdot \sum_{i=1}^{N} \frac{1}{i}
\end{aligned}
$$

Analysis (contd.)

Analyzing one coupon collection problem (contd.)

$$\begin{aligned} &= \sum_{i=1}^{N} \frac{N}{N-i+1} \\ &= N \cdot \sum_{i=1}^{N} \frac{1}{i} \\ &= N \cdot H_N \end{aligned}$$

Analysis (contd.)

Analyzing one coupon collection problem (contd.)

$$
\begin{aligned}
&= \sum_{i=1}^{N} \frac{N}{N-i+1} \\
&= N \cdot \sum_{i=1}^{N} \frac{1}{i} \\
&= N \cdot H_N \\
&= N \cdot \ln N + \Theta(N),
\end{aligned}
$$

Analysis (contd.)

Analyzing one coupon collection problem (contd.)

$$
\begin{aligned}
&= \sum_{i=1}^{N} \frac{N}{N-i+1} \\
&= N \cdot \sum_{i=1}^{N} \frac{1}{i} \\
&= N \cdot H_N \\
&= N \cdot \ln N + \Theta(N), \text{ since } H_N = \ln N + \Theta(1)
\end{aligned}
$$