COL 758: Advanced Algorithms

Lecture 01

Course Policy

- Programming projects - 10%
  (group of size at most two)

- Assignments - 20%
  (must be typed in Latex)
  (group of size at most two)

- Exams - 30% + 35%

- Attendance - 5%

Audit/Credit Pass Criteria

- **Audit:** 35% in all five components (Project, Assignments, Minor, Major, Attendance).

- **Credit:** 30% of course total.

Prerequisites

Courses: COL351/MTL342/COL702

Understanding of:

- Basic probability (random variable, expected value, etc.)
- Knowledge of graphs
- Dynamic programming / Greedy algorithms / Divide and Conquer

What we will study?

1. Randomized Algorithms

Simple and Efficient algorithms
using
Randomness

2. Geometric Algorithms

Algorithms for problems involving
Geometric setting

Eg. Minimum pairwise distance

3. Linear Programming (LP) based Algorithms

Finding approximate solutions to
difficult problems using
LP (linear program)

4. Online Algorithms

Algorithms that work when data is generated
in an on-line manner

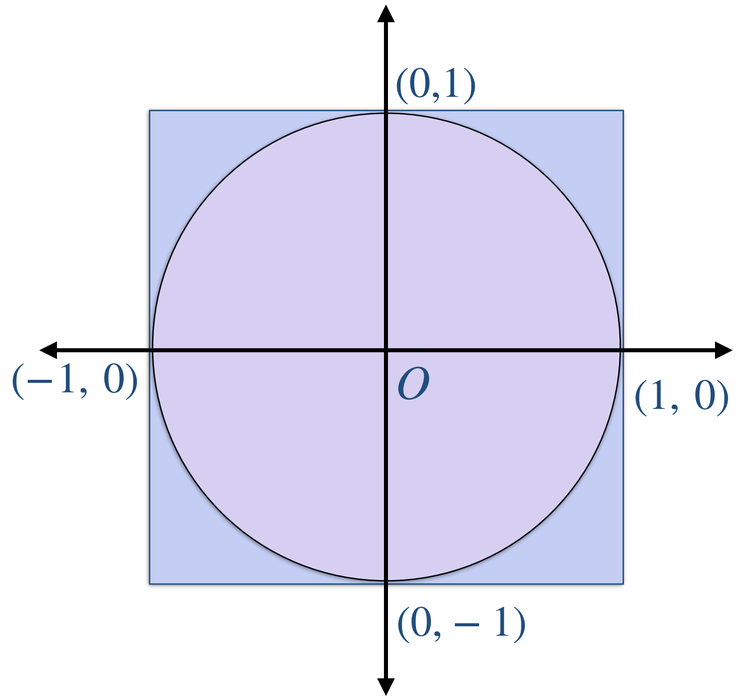
Eg. Google Ad-words

Other concepts (tentative): Gradient descent, streaming algorithms, etc.

I. Randomized Algorithm: Approximating the value of "π"

Generate n random 2D points in square $[-1,1] \times [-1,1]$.

n_{circle} = no. of points in circle of radius one around origin.

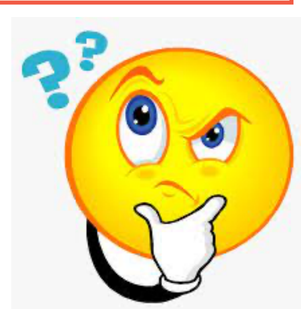


$$\text{Expected}(n_{circle}) = \frac{n \times (area\ of\ circle)}{area\ of\ square} = \frac{n\pi}{4}$$

Question: What is probability that

$$\left| \pi - \frac{4 \cdot n_{circle}}{n} \right| > 10^{-4} \ ?$$

Output: $\frac{4 \cdot n_{circle}}{n}$ as approximation to π



I. Randomized Algorithm: Randomized Quick Sort

Rand-Quick-Sort(L)

Choose pivot x to be Uniformly Random element from list L

Initialise L_1 , L_2 to be empty lists

For each $(y \in L - \{x\})$:

If $(y \leqslant x)$ **then** L_1.append(x)

If $(y > x)$ **then** L_2.append(x)

Return Rand-Quick-Sort(L_1) ∘ x ∘ Rand-Quick-Sort(L_2)

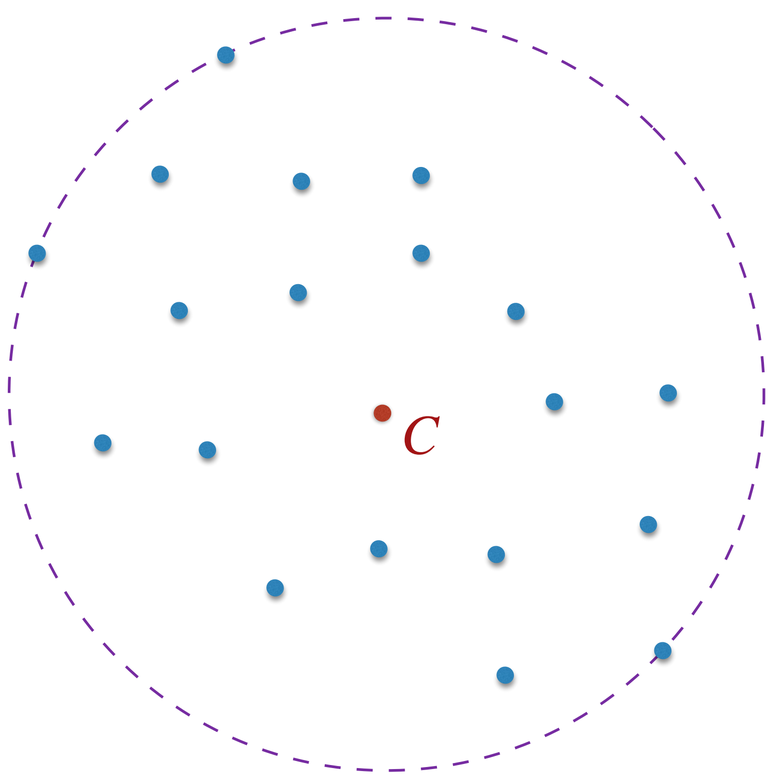
Homework: Prove that expected time-complexity of the algorithm is $O(n \log n)$.

II. Geometric Algorithm: Smallest Enclosing Circle

Given: Set of n points in 2D plane - $(x_1, y_1), \ldots, (x_n, y_n)$.

Output: Circle of minimum radius that contains all points.

Or, find a centre $C = (x_0, y_0)$ for which $\max_{1 \le i \le n} \left\{ \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2} \right\}$ is least.



Naive Algorithm:
Try all (n choose 3 points) and (n choose 2) points.

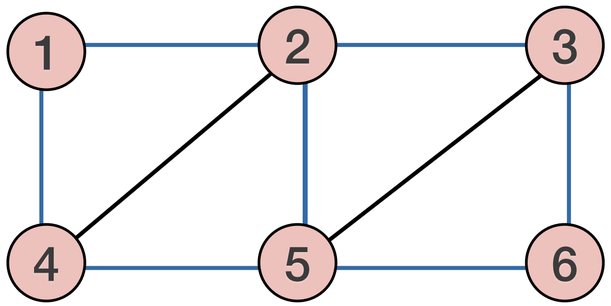
Time: $O(n^4)$

Homework: Think about a natural randomized algorithm!

III. LP Based Algorithm: Approximate vertex cover

Vertex cover: A subset $W \subseteq V$ such that
for each $(a, b) \in E$, an end-point of (a, b) lies in W.

Example:



Goal: Find a 2-approximation to smallest vertex-cover.

- Fact: Computing smallest vertex-cover is NP-hard.

Integer Program (IP) Formulation

Let $(y_1, \ldots, y_n)$ be a vector.

Add binary constraint $y_i \in \{0,1\}$.

For each edge $(i, j) \in E$:

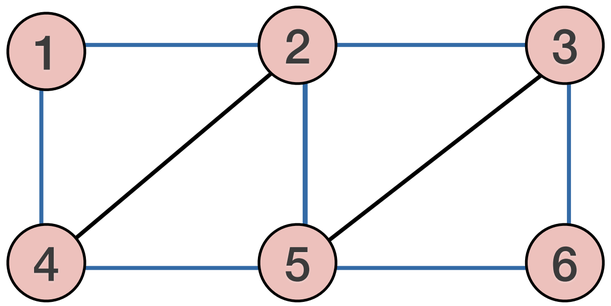
Add constraint: $y_i + y_j \geq 1$

Optimize: $\min_{(y_1,\ldots,y_n)\in\{0,1\}^n} \left(\sum_{i=1}^{n} y_i\right)$

III. LP Based Algorithm: Approximate vertex cover

Vertex cover: A subset $W \subseteq V$ such that for each $(a, b) \in E$, an end-point of (a, b) lies in W.

Example:



Goal: Find a 2-approximation to smallest vertex-cover.

- Fact: Computing smallest vertex-cover is NP-hard.

Linear Program (LP) Formulation

Let $(y_1, \ldots, y_n)$ be a vector.

Add real constraint $y_i \in [0,1]$.

For each edge $(i, j) \in E$:

Add constraint: $y_i + y_j \geq 1$

Optimize: $\min_{(y_1,\ldots,y_n)\in\{0,1\}^n} \left(\sum_{i=1}^{n} y_i\right)$

Step 1. Solve linear program (LP).

Step 2. Transform real $y_i's$ to binary $y_i's$.