CSCI3160 Design and Analysis of Algorithms (2025 Fall)

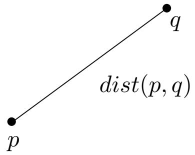
Approximation Algorithms 4: *k*-Center

Instructor: Xiao Liang[1]

Department of Computer Science and Engineering
The Chinese University of Hong Kong

[1] These slides are primarily based on materials prepared by Prof. Yufei Tao (please refer to Prof. Tao's version from 2024 Fall for the original content). Some modifications have been made to better align with this year's teaching progress, incorporating student feedback, in-class interactions, and my own teaching style and research perspective.

Given 2D points p and q, we use $dist(p, q)$ to represent their Euclidean distance.



In this lecture, we will make the assumption that $dist(p, q)$ can be computed in polynomial time.

P = a set of n points in 2D space.

Given a point $p \in P$, define its **distance** to a subset $C \subseteq P$ as

$$dist_C(p) \quad = \quad \min_{c \in C} dist(p, c).$$

The **penality** of C is

$$pen(C) \quad = \quad \max_{p \in P} dist_C(p).$$

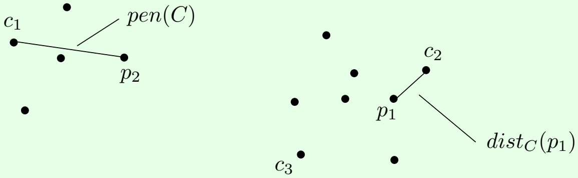
The k-Center Problem: Find a subset $C \subseteq P$ with size $|C| = k$ that has the smallest penalty.

Example:

P = the set of black points

$k = 3$

$C = \{c_1, c_2, c_3\}$



Application 1: Emergency Facility Placement

Scenario: A city plans to build k emergency facilities (e.g., hospitals, fire stations).

Goal:

- Minimize the maximum response time to any area in the city.
- Ensure that every resident is as close as possible to at least one facility.

Model:

- Points: population centers or demand locations.
- Distance: travel time or road distance.

Application 2: Data Clustering

Scenario: In machine learning or data mining, you want to group data into k compact clusters.

Goal:

- Assign each data point to its nearest cluster center.
- Minimize the largest distance between any point and its assigned center.

Use Case:

- Prototype selection in large datasets.
- Reducing latency in content delivery networks.

The problem is NP-hard.

- No one has found an algorithm solving the problem in time polynomial in n and k.
- Such algorithms cannot exist if $\mathcal{P} \neq \mathcal{NP}$.

$\mathcal{A}$ = an algorithm that, given any legal input P, returns a subset of P with size k.

Denote by OPT_P the smallest penalty of all subsets $C \subseteq P$ satisfying $|C| = k$.

> $\mathcal{A}$ is a **ρ-approximate algorithm** for the k-center problem if, for any legal input P, $\mathcal{A}$ can return a set C with penalty at most $\rho \cdot OPT_P$.

The value ρ is the **approximation ratio**.
We say that $\mathcal{A}$ achieves an approximation ratio of ρ.

Approximation Algorithm with $\rho = 2$

Consider the following greedy algorithm:

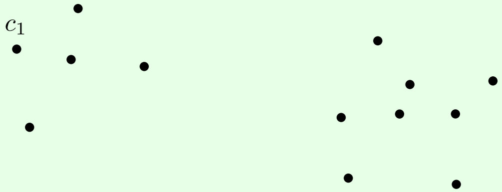
Input: P

1. $C \leftarrow \emptyset$
2. add to C an arbitrary point in P
3. **for** $i = 2$ to k **do**
4. $\quad p \leftarrow$ a point in P with the maximum $dist_C(p)$
5. $\quad$ add p to C
6. return C

The algorithm can be easily implemented in polynomial time.

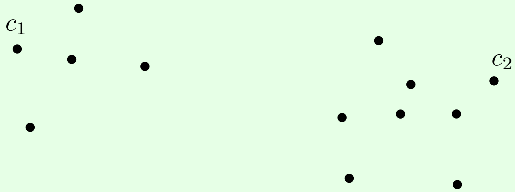
Later, we will prove that the algorithm is 2-approximate.

Example: $k = 3$



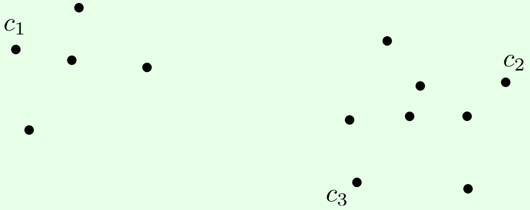
Initially, $C = \{c_1\}$

Example: $k = 3$



After a round, $C = \{c_1, c_2\}$

Example: $k = 3$



After another round, $C = \{c_1, c_2, c_3\}$

Proof of Approximation Guarantee

The approximation guarantee is established by the following theorem.

Theorem 1: The algorithm returns a set C with $pen(C) \leq 2 \cdot OPT_P$.

Next, we prove this theorem.

Proof: Let $C^* = \{c_1^*, c_2^*, ..., c_k^*\}$ be an optimal solution, i.e., $pen(C^*) = OPT_P$.

For each $i \in [1, k]$, define P_i^* as the set of points $p \in P$ satisfying

$$dist(p, c_i^*) \leq dist(p, c_j^*)$$

for any $j \neq i$.

(Intuitively, P_i^* is the set of points clustered around the i-th center c_i^*.)

Observation 1:
For any point $p \in P_i^*$, $dist(p, c_i^*) = dist_{C^*}(p) \leq pen(C^*)$.

Let C_{ours} be the output of our algorithm.

Case 1: C_{ours} has a point in each of $P_1^*, P_2^*, ..., P_k^*$.

Consider any point $p \in P$. Suppose that $p \in P_i^*$ for some $i \in [1, k]$.
Let c be a point in $C_{ours} \cap P_i^*$. It holds that:

$$\begin{aligned} dist_{C_{ours}}(p) &\leq dist(c, p) && \text{(by def. of distance)} \\ &\leq dist(c, c_i^*) + dist(c_i^*, p) && \text{(by triangle inequality)} \\ &\leq 2 \cdot pen(C^*) && \text{(by Observation 1 in the last slide)} \end{aligned}$$

Therefore:

$$pen(C_{ours}) = \max_{p \in P} dist_{C_{ours}}(p) \leq 2 \cdot pen(C^*).$$

Case 2: C_{ours} has no point in at least one of $P_1^*, ..., P_k^*$. Hence, one of $P_1^*, ..., P_k^*$ — say P_i^* — must cover at least two points c_a and c_b of C_{ours}. It thus follows that

$$\begin{aligned} dist(c_a, c_b) &\leq dist(c_a, c_i^*) + dist(c_b, c_i^*) \quad \text{(by triangle inequality)} \\ &\leq 2 \cdot pen(C^*). \quad \text{(by Observation 1)} \end{aligned}$$

Next, we prove:

> **Claim 1:** For any point $p \in P$, $dist_{C_{ours}}(p) \leq dist(c_a, c_b)$.

Note that **Claim 1** implies $pen(C_{ours}) \leq 2 \cdot pen(C^*)$.

This finishes the proof of **Theorem 1** (modulo **Claim 1** which we prove next).

Proof of Claim 1

W.l.o.g., assume that c_b was picked after c_a by our algorithm. Consider the moment right before c_b was picked. At that moment, the set C maintained by our algorithm was a proper subset of C_{ours}.

From the fact that c_b was the next point picked, we know $dist_C(p) \leq dist_C(c_b)$ for every $p \in P$. /* Here, we utilize the greedy nature of our algorithm*/

Because $c_a \in C$, it holds that $dist_C(c_b) \leq dist(c_a, c_b)$.

The lemma then follows because

$$dist_{C_{ours}}(p) \leq dist_C(p) \leq dist_C(c_b) \leq dist(c_a, c_b).$$

This finishes the proof of **Claim 1**. $\square$