COMP4128 Week 07 Tutorial

Yifan He

z5173587@unsw.edu.au
`https://github.com/hharryyf/COMP4128-23T3-tutoring`

Contest 2 statistics

- Mean: 141
  - 9x 300
  - 10x 250
  - 13x 200
  - 7x 150
  - 15x 100
  - 13x 50
  - 11x 0
- Reminder: for those who got 0 or 50, you can email me if you have specific questions related to the course

Outline

- Shortest path revision
- Roads in Berland
- Weights Distributing
- PS5 Hints given by email

Shortest path revision

Algorithm	Big-O	**Source**	**Requirement**
BFS	O(E)	Single	edge weights are all equal
Dijkstra	Elog(E)	Single	edge weights non-negative
Bellman-ford (SPFA)	VE	Single	no negative cycle system of different constraints
Floyd	V^3	All-pair	no negative cycle

- Given a weighted undirected graph, each edge is represented as (u, v, w), is $e(u_i, v_i, w_i)$ on some shortest path between s and t?

Shortest path revision

Algorithm	Big-O	**Source**	**Requirement**
BFS	O(E)	Single	edge weights are all equal
Dijkstra	Elog(E)	Single	edge weights non-negative
Bellman-ford (SPFA)	VE	Single	no negative cycle system of different constraints
Floyd	V^3	All-pair	no negative cycle

- Given a weighted undirected graph, each edge is represented as (u, v, w), is $e(u_i, v_i, w_i)$ on some shortest path between s and t?
- $dist(s, u_i) + dist(v_i, t) + w_i = dist(s, t)$, or
- $dist(s, v_i) + dist(u_i, t) + w_i = dist(s, t)$

Roads in Berland [1]

Given a weighted complete graph with $|V|$ ($|V| \leq 300$) vertices. You need to process k ($k \leq 300$) operations. Each operation is going to add a new road between two cities u and v with weight w. After each operation, return the sum of the shortest distance between all pairs of vertices.

[1] `https://codeforces.com/contest/25/problem/C`

Roads in Berland

Naive Approach

- Run Floyd algorithm after each edge addition
- k runs, each runs takes $O(|V|^3)$
- Time complexity: $O(k \cdot |V|^3)$
- Too slow

Roads in Berland

Analysis

- Think how the edge $e(u, v, w)$ affect the shortest path from a to b
- 2 cases:
  - $dist(a, b)$ is not going to be affected
  - $dist(a, b)$ is going to be affected
- In the second case, $e(u, v, w)$ is on the shortest path between a and b in the new graph
-

$$dist(a, b) = min \begin{cases} dist(a, b) \\ dist(a, u) + w + dist(v, b) \\ dist(a, v) + w + dist(u, b) \end{cases} \quad (1)$$

Roads in Berland

- $O(|V|^2)$ per modification
- $O(k \cdot |V|^2)$ in total

Roads in Berland

Demo

Weights Distributing

Given an undirected graph with V ($|V| \leq 2e5$) vertices and E ($|E| \leq 2e5$) edges. Given a list of length $|E|$ of positive integers $p_1 \dots p_{|E|}$. Assign the edges of the graph with weights from this list (1-1 mapping). What is the shortest distance from a to b and then from b to c?

Example

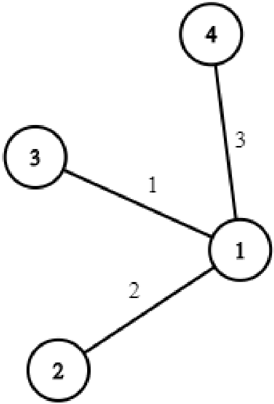


Figure: a=2,b=3,c=4,p=[1,2,3]

Weights Distributing

Simplified problem 1

- The graph is a line
- There are k_1 edges between a and b
- k_2 edges between b and c
- Suppose the array p is $[p_1, p_2, ..., p_{|E|}]$, $k_1 + k_2 \leq |E|$
- How would you assign the edges?

Weights Distributing

Simplified problem 1

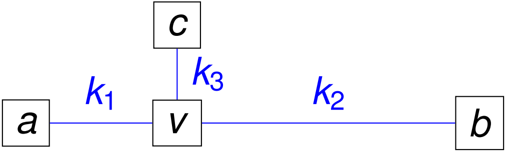
- The graph is a line
- There are k_1 edges between a and b
- k_2 edges between b and c
- Suppose the array p is $[p_1, p_2, ..., p_{|E|}]$, $k_1 + k_2 \leq |E|$
- How would you assign the edges?

- Smallest $k_1 + k_2$ elements of p to the edges between a and c

Weights Distributing

Simplified problem 2

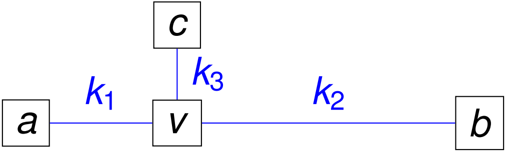
- There are k_1 edges between a and v
- k_2 edges between b and v
- k_3 edges between c and v
- How would you assign the edges?



Weights Distributing

Simplified problem 2

- There are k_1 edges between a and v
- k_2 edges between b and v
- k_3 edges between c and v
- How would you assign the edges?

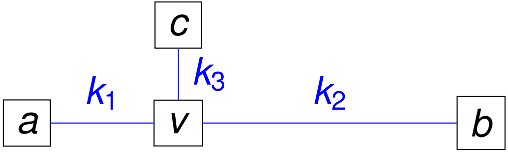


- Smallest $k_1 + k_2 + k_3$ edges

Weights Distributing

Simplified problem 2

- There are k_1 edges between a and v
- k_2 edges between b and v
- k_3 edges between c and v
- How would you assign the edges?

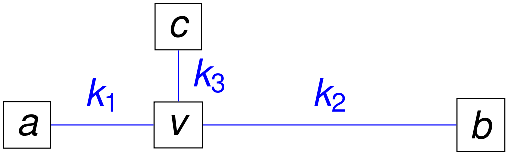


- Smallest $k_1 + k_2 + k_3$ edges
- **Also**, smallest k_2 edges between b and v

Weights Distributing

General problem

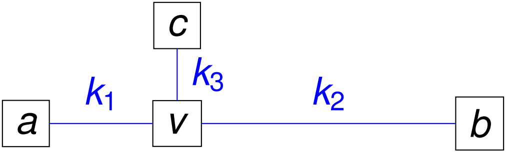
- We only need to consider a substructure as follows



- Sort the array p in non-decreasing order
- Iterate through $1 \ldots |V|$ and let it be vertex v
- $k_1 = dist(a, v)$, $k_2 = dist(b, v)$, $k_3 = dist(c, v)$

Weights Distributing

General problem



- For each $v \in V$
  - $k_1 = dist(a, v)$, $k_2 = dist(b, v)$, $k_3 = dist(c, v)$
  - Let $sum(i)$ be sum of $p[1 \ldots i]$
  - Minimum: $sum(1, k_2) + sum(1, k_1 + k_2 + k_3)$
- We can precompute $dist(a, v)$, $dist(b, v)$, and $dist(c, v)$ for all v with BFS

Weights Distributing

Demo