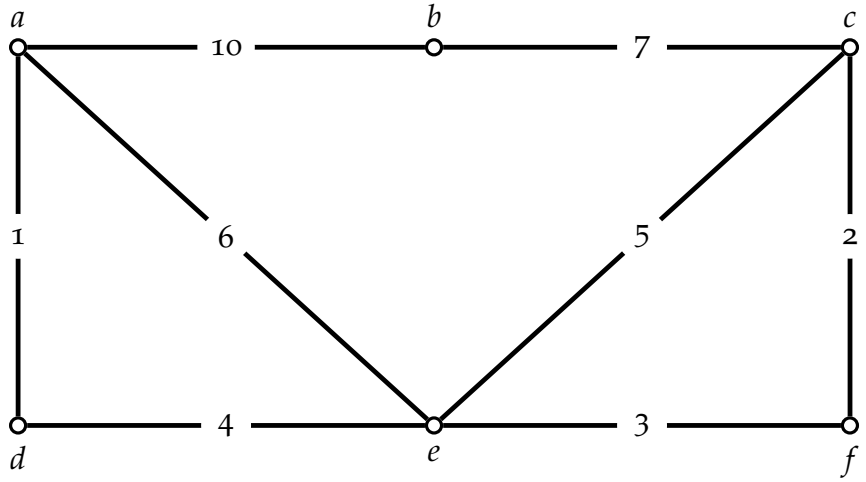
Warm-up

Problem 1. Consider Dijkstra's shortest path algorithm for undirected graphs. What changes (if any) do we need to make to this algorithm for it to work for directed graphs and maintain its running time?

Problem 2. Consider the following weighted undirected graph G:



Your task is to compute a minimum weight spanning tree T of G:

a) Which edges are part of the MST?

b) In which order does Kruskal's algorithm add these edges to the solution.

c) In which order does Prim's algorithm (starting from a) add these edges to the solution.

Problem solving

Problem 3. Let $G = (V, E)$ be an undirected graph with edge weights $w : E \to R^+$. For all $e \in E$, define $w_1(e) = \alpha w(e)$ for some $\alpha > 0$, $w_2(e) = w(e) + \beta$ for some $\beta > 0$, and $w_3(e) = w(e)^2$.

a) Suppose p is a shortest s-t path for the weights w. Is p still optimal under w_1? What about under w_2? What about under w_3?

b) Suppose T is minimum weight spanning tree for the weights w. Is T still optimal under w_1? What about under w_2? What about under w_3?

Problem 4. It is not uncommon for a given optimization problem to have multiple optimal solutions. For example, in an instance of the shortest s-t path problem, there could be multiple shortest paths connecting s and t. In such situations, it may be desirable to break ties in favor of a path that uses the fewest edges.

Show how to reduce this problem to a standard shortest path problem. You can assume that the edge lengths ℓ are positive integers.

a) Let us define a new edge function $\ell'(e) = M\ell(e)$ for each edge e. Show that if P and Q are two s-t paths such that $\ell(P) < \ell(Q)$ then $\ell'(Q) - \ell'(P) \geq M$.

b) Let us define a new edge function $\ell''(e) = M\ell(e) + 1$ for each edge e. Show that if P and Q are two s-t paths such that $\ell(P) = \ell(Q)$ but P uses fewer edges than Q then $\ell''(P) < \ell''(Q)$.

c) Show how to set M in the second function so that the shortest s-t path under ℓ'' is also shortest under ℓ and uses the fewest edges among all such shortest paths.

Problem 5. Consider the following generalization of the shortest path problem where in addition to edge lengths, each vertex has a cost. The objective is to find an s-t path that minimizes the total length of the edges in the path plus the cost of the vertices in the path. Design an efficient algorithm for this problem.

Problem 6. Consider the IMPROVING-MST algorithm for the MST problem.

```
1: function IMPROVING-MST(G, w)
2:     T ← some spanning tree of G
3:     for e ∈ E \ T do
4:         T ← T + e
5:         C ← unique cycle in T
6:         f ← heaviest edge in C
7:         T ← T − f
8:     return T
```

Prove its correctness and analyze its time complexity. To simplify things, you can assume the edge weights are distinct.

Problem 7. Consider the REVERSE-MST algorithm for the MST problem.

```
1: function REVERSE-MST(G, w)
2:     Sort edges in decreasing weight w
3:     T ← E
4:     for e ∈ E in decreasing weight do
5:         if T − e is connected then
6:             T ← T − e
7:     return T
```

Prove its correctness and analyze its time complexity. To simplify things, you can assume the edge weights are distinct.

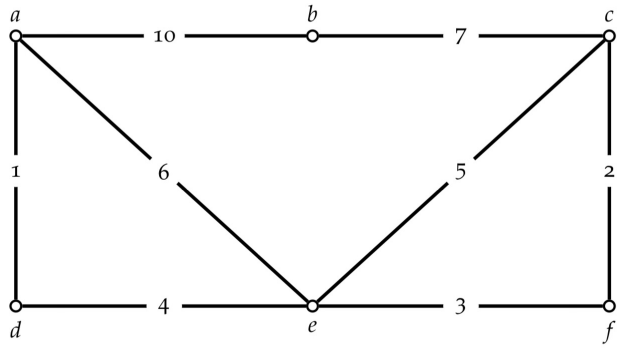
Problem 8. A computer network can be modeled as an undirected graph $G = (V, E)$ where vertices represent computers and edges represent physical links between computers. The maximum transmission rate or *bandwidth* varies from link to link; let $b(e)$ be the bandwidth of edge $e \in E$. The bandwidth of a path P is defined as the minimum bandwidth of edges in the path, i.e., $b(P) = \min_{e \in P} b(e)$.

Suppose we number the vertices in $V = \{v_1, v_2, \ldots, v_n\}$. Let $B \in R^{n \times n}$ be a matrix where B_{ij} is the maximum bandwidth between v_i and v_j. Give an algorithm for computing B. Your algorithm should run in $O(n^3)$ time.

Problem 1. Consider Dijkstra's shortest path algorithm for undirected graphs. What changes (if any) do we need to make to this algorithm for it to work for directed graphs and maintain its running time?

- We just need to consider about the direction of edge for current vertex

Problem 2. Consider the following weighted undirected graph G:



Your task is to compute a minimum weight spanning tree T of G:

a) Which edges are part of the MST?

b) In which order does Kruskal's algorithm add these edges to the solution.

c) In which order does Prim's algorithm (starting from a) add these edges to the solution.

(b) ad, cf, ef, de, bc,

(c) ad, de, ef, fc, bc

Problem 3. Let $G = (V, E)$ be an undirected graph with edge weights $w : E \to R^+$. For all $e \in E$, define $w_1(e) = \alpha w(e)$ for some $\alpha > 0$, $w_2(e) = w(e) + \beta$ for some $\beta > 0$, and $w_3(e) = w(e)^2$.

a) Suppose p is a shortest s-t path for the weights w. Is p still optimal under w_1? What about under w_2? What about under w_3?

b) Suppose T is minimum weight spanning tree for the weights w. Is T still optimal under w_1? What about under w_2? What about under w_3?

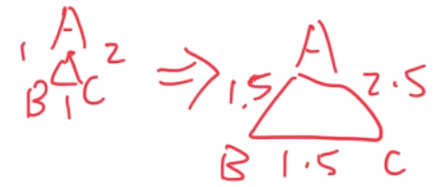
$\alpha(e_1+e_2)=\alpha e_3$

$w_1(e)$

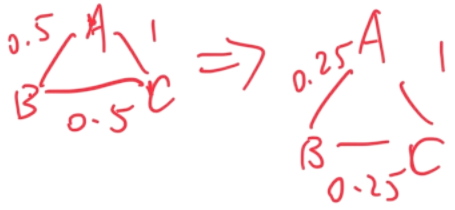
(a) . Yes, since $\alpha>0 \Rightarrow \alpha w(e)$ is still the smallest weight

affect multiple edge will not change result

. $w_2(e)$: No,



. $w_3(e)$: No



affect multiple edge will change result

$e_1+\beta+e_2+\beta > e_3+\beta$

$e_1 \cdot e_1 + e_2 \cdot e_2 \geq e_3 \leq e_3$

(b) Yes for w_1, w_2, w_3

Since its MST, the relative weight between each edge does not change

Problem 4. It is not uncommon for a given optimization problem to have multiple optimal solutions. For example, in an instance of the shortest s-t path problem, there could be multiple shortest paths connecting s and t. In such situations, it may be desirable to break ties in favor of a path that uses the fewest edges.

Show how to reduce this problem to a standard shortest path problem. You can assume that the edge lengths ℓ are positive integers.

a) Let us define a new edge function $\ell'(e) = M\ell(e)$ for each edge e. Show that if P and Q are two s-t paths such that $\ell(P) < \ell(Q)$ then $\ell'(Q) - \ell'(P) \geq M$.

b) Let us define a new edge function $\ell''(e) = M\ell(e) + 1$ for each edge e. Show that if P and Q are two s-t paths such that $\ell(P) = \ell(Q)$ but P uses fewer edges than Q then $\ell''(P) < \ell''(Q)$.

c) Show how to set M in the second function so that the shortest s-t path under ℓ'' is also shortest under ℓ and uses the fewest edges among all such shortest paths.

(a) $\ell(Q) = X$ $\ell(P) = Y$ [WTS: $MX - MY \geq M$ for some M]

. given $X > Y \Rightarrow X - Y > 0 \Rightarrow X - Y \geq 1$ because $X, Y > 0$

. if $M \geq 0$ then $MX - MY \geq M$

(b) each edge increase 1 ⇒ more edges increase more weight

(c) Suppose $L(P) > L(Q)$

$L(P) - L(Q) \geq 1$ # because $L(P) > 0$ and $L(Q) > 0$ and $L(P) > L(Q)$

$L(P) \geq 1 + L(Q)$

① $ML(P) \geq \underline{M + ML(Q)}$ # $L''(Q) = 1 + ML(Q)$

$\hookrightarrow M + ML(Q) > |Q| + ML(Q)$

$\geq 1 + ML(Q) = L''(Q)$

$M > |Q|$

其中 M 是所有 edge，$|Q|$ 是 the number of edges used in the path Q

② $ML(P) \leq |P| + ML(P) \leq L''(P)$

①②: $L''(P) \geq ML(P) \geq M + ML(Q) > L''(Q)$

Problem 5. Consider the following generalization of the shortest path problem where in addition to edge lengths, each vertex has a cost. The objective is to find an s-t path that minimizes the total length of the edges in the path plus the cost of the vertices in the path. Design an efficient algorithm for this problem.

· Using MST

· add vertex weight to edge, and calculate

Problem 6. Consider the IMPROVING-MST algorithm for the MST problem.

```
1: function IMPROVING-MST(G, w)
2:     T ← some spanning tree of G
3:     for e ∈ E \ T do
4:         T ← T + e
5:         C ← unique cycle in T
6:         f ← heaviest edge in C
7:         T ← T − f
8:     return T
```

O(m-T) = O(m)

O(1)

O(n)

O(n)

O(1)

⇒ O(mn) ≥ O(n²)

Prove its correctness and analyze its time complexity. To simplify things, you can assume the ~~edge weights are distinct.~~

Problem 7. Consider the REVERSE-MST algorithm for the MST problem.

```
1: function REVERSE-MST(G, w)
2:     Sort edges in decreasing weight w
3:     T ← E
4:     for e ∈ E in decreasing weight do
5:         if T − e is connected then
6:             T ← T − e
7:     return T
```

Prove its correctness and analyze its time complexity. To simplify things, you can assume the edge weights are distinct.

Problem 8. A computer network can be modeled as an undirected graph $G = (V, E)$ where vertices represent computers and edges represent physical links between computers. The maximum transmission rate or *bandwidth* varies from link to link; let $b(e)$ be the bandwidth of edge $e \in E$. The bandwidth of a path P is defined as the minimum bandwidth of edges in the path, i.e., $b(P) = \min_{e \in P} b(e)$.

Suppose we number the vertices in $V = \{v_1, v_2, \ldots, v_n\}$. Let $B \in R^{n \times n}$ be a matrix where B_{ij} is the maximum bandwidth between v_i and v_j. Give an algorithm for computing B. Your algorithm should run in $O(n^3)$ time.

- Since we want to find max bandwidth
- we can just find the max weighted edges instead of min weighted edge in Prim's algorithm
- 我们有 n^2 个 vertices pair (v_i, v_j)
  每个 vertices pair 找到 max weighted path take $O(n)$
- In total $O(n^3)$

不用考虑 (x, y) 于 (y, x) 重复的问题