Problem Set 7

This problem set is due **at 11:59pm** on **Thursday, April 9, 2015.**

Each submitted solution should start with your name, the course number, the problem number, your recitation section, the date, and the names of any students with whom you collaborated.

Exercise 7-1. Read CLRS, Sections 26.1-26.3.

Exercise 7-2. Exercise 26.1-2.

Exercise 7-3. Exercise 26.1-4.

Exercise 7-4. Exercise 26.2-4.

Exercise 7-5. Exercise 26.2.6.

Problem 7-1. Maximum Flow in a Dynamic Network [20 points]

In this problem, you will design an algorithm that takes the following inputs:

- A flow network $F = (G, c)$, where $G = (V, E)$ is a graph with source vertex s and target vertex t, and c is a capacity function mapping each directed edge of G to a nonnegative integer;
- A maximum flow f for F; and
- A triple (u, v, r), where u and v are vertices of G and r is a nonnegative integer $\neq c(u, v)$.

The algorithm should produce a maximum flow for flow network $F' = (G, c')$, where c' is identical to c except that $c'(u, v) = r$. The algorithm should run in time $O(k \cdot (V + E))$, where $|c(u, v) - r| = k$. The algorithm should behave differently depending on whether $r > c(u, v)$ or $r < c(u, v)$.

(a) [4 points] Start by proving the following basic, general results about flow networks:

1. Increasing the capacity of a single edge (u, v) by 1 can result in an increase of at most 1 in the max flow.
2. Increasing the capacity of a single edge (u, v) by a positive integer k can result in an increase of at most k in the max flow.
3. Decreasing the capacity of a single edge (u, v) by 1 can result in a decrease of at most 1 in the max flow.
4. Decreasing the capacity of a single edge (u, v) by a positive integer k can result in a decrease of at most k in the max flow.

$s_1, s_2 \cdots s_n$ are all intermidiate vertix.

$t_1, t_2 \cdots t_m$

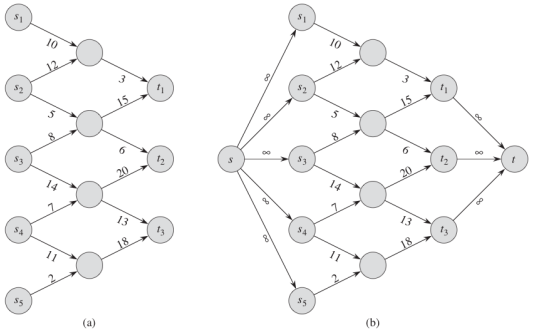
so: for s: $\Sigma_{in} = \Sigma_{out}$.

$s.out.$

for t: $\Sigma_{in} = \Sigma_{out}$.

$|f| = t.in.$

26.1-2
Extend the flow properties and definitions to the multiple-source, multiple-sink problem. Show that any flow in a multiple-source, multiple-sink flow network corresponds to a flow of identical value in the single-source, single-sink network obtained by adding a supersource and a supersink, and vice versa.



26.1-4
Let f be a flow in a network, and let α be a real number. The ***scalar flow product***, denoted αf, is a function from $V \times V$ to $\mathbb{R}$ defined by

$$(\alpha f)(u, v) = \alpha \cdot f(u, v) .$$

Prove that the flows in a network form a ***convex set***. That is, show that if f_1 and f_2 are flows, then so is $\alpha f_1 + (1 - \alpha) f_2$ for all α in the range $0 \le \alpha \le 1$.

$f_1 \Rightarrow$

1. for any u v.

$f_1(u,v) \le c(u,v)$

$f_2(u,v) \le c(u,v)$.

+ $\alpha f_1 \le \alpha c(u,v)$

$(1-\alpha) f_2 \le (1-\alpha) c(u,v)$.

$f_3 = \alpha f_1 + (1-\alpha) f_2 \le c(u,v)$. 2. [capacity constrait.]

1. flow conservation. in f_1.

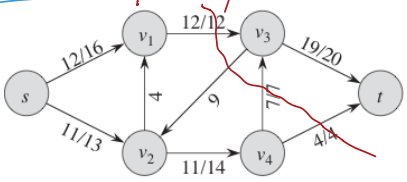
$\sum_{v \in V} (f_3) = \alpha \sum_{v \in V} f_1(u,v) + (1-\alpha) \sum_{v \in V} f_2(u,v)$

$= \alpha \sum_{v \in V} f_1(v,u) + (1-\alpha) \sum_{v \in V} f_2(v,u)$

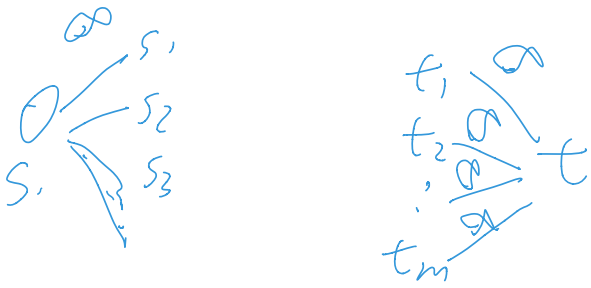
$= \sum_{v \in V} \alpha f_1(v,u) + (1-\alpha) f_2(v,u)$

2.

to prove capacity constrait and flow conservation.



26.2-4
In the example of Figure 26.6, what is the minimum cut corresponding to the maximum flow shown? Of the augmenting paths appearing in the example, which one cancels flow?



26.2-6
Suppose that each source s_i in a flow network with multiple sources and sinks produces exactly p_i units of flow, so that $\sum_{v \in V} f(s_i, v) = p_i$. Suppose also that each sink t_j consumes exactly q_j units, so that $\sum_{v \in V} f(v, t_j) = q_j$, where $\sum_i p_i = \sum_j q_j$. Show how to convert the problem of finding a flow f that obeys these additional constraints into the problem of finding a maximum flow in a single-source, single-sink flow network.

(1) if (u,v) is on ~~minimum~~ any capacity cut, $+1 \Rightarrow +1 \Rightarrow$ 不变
max flow $+1$

if not flow $\rightarrow$.

(2) while (k) times $\leq k$.

(3) same idea.

b) $r > c(u,v)$.

repeat k times.

if find a augmenting path in the residual network.

augument the ~~path~~ graph.

else

return.

augments at least add flow 1, add most k augments, add k flow,

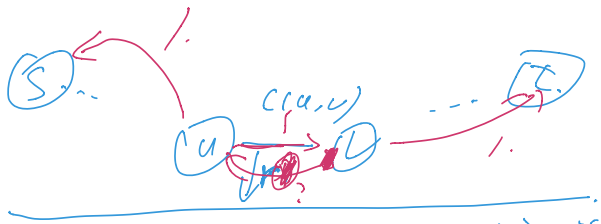
so loop most k times. use BFS to find an augmenting path

in $O(V+E) \Rightarrow O(k\cdot(V+E))$.

c) $r < c(u,v)$.

从r开始找 augmenting path,

~~不~~找不到 decreasing path.



如果r小, other path 可能找不到.

不是 $s-u$, $v-t$ 3.

if $f(u,v) \leq r$. so no need.

to change.

if $f(u,v) > r$. decrese $f(u,v)$

by one until reach r.

before that, we should ~~ensure~~ ensure the conservation constraint

① decrese $f(u,v)$ by one.

② DFS from u. if reach s, t, or v. in backward.

+ if reach v, ~~dec~~ decrese the path flow by 1.

- if reach s, the run dfs from v to t, decrease 1.

reach t, ---- v to s, decrease 1.

then run Part b. phase 1.

$O(k\cdot(V+E))$ phase 2.

(b) [8 points] Suppose that $r > c(u, v)$. Describe your algorithm for this case in detail, prove that it works correctly, and analyze its time complexity (in terms of V, E, and k).

(c) [8 points] Suppose that $r < c(u, v)$. Describe your algorithm for this case in detail, prove that it works correctly, and analyze its time complexity.

Problem 7-2. Disjoint Roads [15 points]

A number k of trucking companies, $c_1, \ldots, c_k$, want to use a common road system, which is modeled as a directed graph, for delivering goods from source locations to a common target location. Each trucking company c_i has its own source location, modeled as a vertex s_i in the graph, and the common target location is another vertex t. (All these $k + 1$ vertices are distinct.)

The trucking companies want to share the road system for delivering their goods, but they want to avoid getting in each other's way while driving. Thus, they want to find k edge-disjoint paths in the graph, one connecting each source s_i to the target t. We assume that there is no problem if trucks of different companies pass through a common vertex.

Design an algorithm for the companies to use to determine k such paths, if possible, and otherwise return "impossible".

Problem 7-3. Food Truck Orders [15 points]

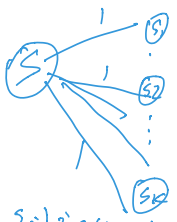
The Miso Good food truck produces a large variety of different lunch menu items. Unfortunately, they can only produce their foods in limited quantities, so they often run out of popular items, making customers sad.

To minimize sadness, Miso Good is implementing a sophisticated lunch-ordering system. Customers text in their acceptable choices before lunch time. Then they can use an algorithm to preassign lunches to customers. Customers who do not get one of their choices should receive a $\$10$ voucher. Miso Good would like to minimize the number of vouchers they give out.

Give an efficient algorithm for Miso Good to assign lunches to customers. In general, suppose that, on a given day, Miso Good has produced m types of food items $b_1, \ldots, b_m$, and the quantity of each type of food item b_j is exactly q_j. Suppose that n customers $a_1, \ldots, a_n$ text in their preferences, where each customer a_i submits a set A_i of one or more acceptable lunch choices. The algorithm should assign each customer either one of his/her choices or a $\$10$ voucher. It should minimize the number of vouchers.

(*Hint:* Model this as a max flow problem.)

7-2.



t

add s. (s, s_i) i∈(1, k) to G, ~~cost~~

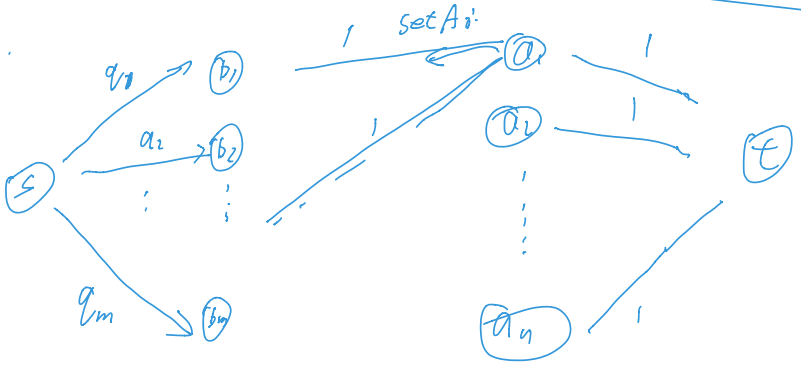
for graph G(V, E), C(e) ⇒ 1.

then compute the maximum flow of graph.

G(V, E, c), if |f| = k, ok, else impossible.

O(f(k))

7-3.



so maximum flow of graph.

~~#~~ · # Edge. m × n.

the maximum |f| = n. (run most n times of Ford-Fulkerson)

so $O(m \cdot n^2)$.

6.046J / 18.410J Design and Analysis of Algorithms
Spring 2015