College of Computer & Information Science
Northeastern University
Fall 2017

CS5800
Algorithms
December 8 2017

CS5800 - Algorithms - Review for final exam

This file includes only a sketch of the solutions. The following solutions are not full and they include only the main ideas.

1. Let $G = (V, E)$ be a flow network with capacity $c : E \to \mathbb{R}_{\geq 0}$.
   For 2 s-t cuts $(S, \overline{S})$ and $(S', \overline{S'})$, we define their union to be $(S \cup S', \overline{S \cup S'})$.
   Show that if $(S, \overline{S})$ and $(S', \overline{S'})$ are both minimal cuts, then their union $(S \cup S', \overline{S \cup S'})$ is also a minimal cut.

   Answer: Let $f_0 : E \to \mathbb{R}$ be a maximal flow for G.
   Since S is a minimal cut and f_0 is a maximal flow, we have:

$$\text{val}(f_0) = \text{cap}(S, \overline{S}) = \sum_{(u,v) \in ((S \times \overline{S}) \cap E)} c(u, v)$$

   On the other hand we have:

$$\text{cap}(S, \overline{S}) = \text{val}(f_0) = f_0(S, \overline{S}) - f_0(\overline{S}, S) \leq f_0(S, \overline{S}) \leq \sum_{(u,v) \in ((S \times \overline{S}) \cap E)} c(u, v) = \text{cap}(S, \overline{S})$$

   It follows that:

$$f_0(S, \overline{S}) = \sum_{(u,v) \in ((S \times \overline{S}) \cap E)} c(u, v)$$

   and:

$$f_0(\overline{S}, S) = 0$$

   We see that we must have $f_0(u, v) = \text{cap}(u, v)$ for any edge of the form $(u, v) \in ((S \times \overline{S}) \cap E)$, and $f_0(u, v) = 0$ for any edge of the form $(u, v) \in ((\overline{S} \times S) \cap E)$.
   Let (u, v) be an edge from $S \cup S'$ to $\overline{S \cup S'}$ (that is $u \in S \cup S'$, $v \in \overline{S \cup S'}$). It follows that $u \in S$ or $u \in S'$ and $v \notin S$ and $v \notin S'$. Assume $u \in S$. The edge (u, v) goes from S to $\overline{S}$ and hence by what we saw $f_0(u, v) = c(u, v)$.
   Let (v, u) be an edge from $\overline{S \cup S'}$ to $S \cup S'$. It follows that $v \notin S$ and $v \notin S'$ and $u \in S$ or $u \in S'$. Assume $u \in S$. The edge (v, u) goes from $\overline{S}$ to S and hence by what we saw $f_0(v, u) = 0$.
   We see that for any edge e from $S \cup S'$ to $\overline{S \cup S'}$ we have:

$$f_0(e) = c(e)$$

   and for any edge e from $\overline{S \cup S'}$ to $S \cup S'$ we have:

$$f_0(e) = 0$$

   It follows that:

$$\text{val}(f_0) = f(S \cup S', \overline{S \cup S'}) - f(\overline{S \cup S'}, S \cup S') = \text{cap}(S \cup S') - 0 = \text{cap}(S \cup S')$$

   and thus by the min cut max flow theorem $S \cup S'$ is a minimal cut.

2. Your company has n different computational tasks $T_1, T_2, ..., T_n$ that it needs to execute. There are 2 computers c_1 and c_2 that can execute these computational tasks. Each of the tasks $T_1, ..., T_n$ can be executed by any of the 2 computers c_1 or c_2.
You know that the costs of the computations are given by (for $1 \leq i \leq n$):

   - The cost of executing task T_i by c_1 is $a_i > 0$.
   - The cost of executing task T_i by c_2 is $b_i > 0$.
   - If T_i and T_j are executed by 2 different computers (clearly $i \neq j$), there is an additional cost of d_{ij}. (The additional cost is the same whether T_i is executed by c_1 and T_j by c_2 or T_i is executed by c_2 and T_j by c_1.)

   Suggest an algorithm to find the best way to divide the tasks between the two computers.

   Answer: We construct a graph $G = (V, E)$ where V is:

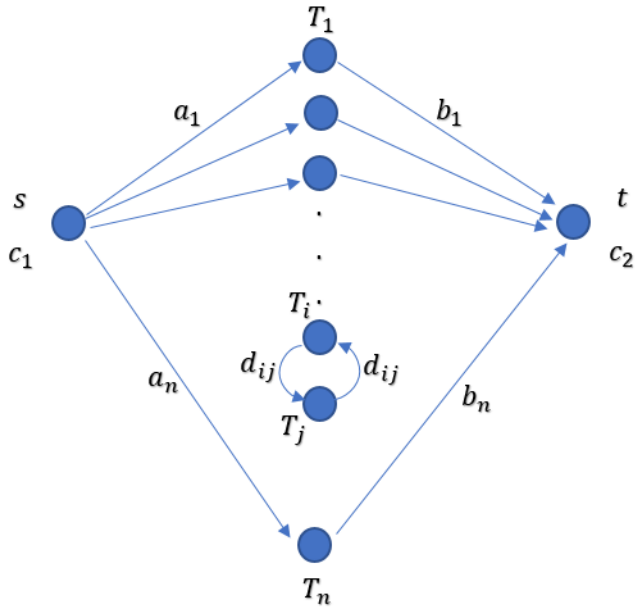
$$V = \{s\} \cup \{T_1, T_2, ..., T_n\} \cup \{t\}$$

   and:

$$E = \{(s, T_i) \mid 1 \leq i \leq n\} \cup \{(T_i, t) \mid 1 \leq i \leq n\} \cup \{(T_i, T_j) \mid 1 \leq i, j \leq n \text{ and } i \neq j\}$$

   (We think of s as c_1 and of t as c_2.)
   We define the capacities of the edges by:

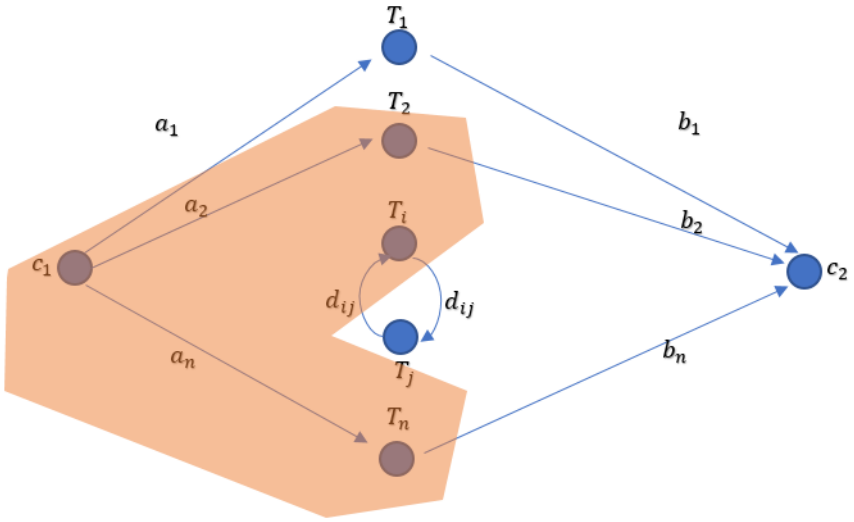
$$w(e) = \begin{cases} a_i, & e = (s, T_i); \\ b_i, & e = (T_i, t); \\ d_{ij}, & e = (T_i, T_j). \end{cases}$$



Let $(S, \overline{S})$ be an $s-t$ cut. We have:

$$\text{cap}(S, \overline{S}) = \sum_{T_i \notin S} a_i + \sum_{T_i \in S} b_i + \sum_{T_i \in S \text{ and } T_j \notin S} d_{ij}$$

We see that the capacity of $(S, \overline{S})$ is exactly the cost of performing the tasks $\{T_i \mid T_i \notin S\}$ in c_1 and the tasks $\{T_i \mid T_i \in S\}$ in c_2.



It follows that in order to find the most efficient way to perform $T_1, ..., T_n$ we need to find a minimum cut.

3. Let $A_1, ..., A_n$ be n sets (not necessarily disjoint sets. It might be that $A_i \cap A_j \neq \emptyset$), and let $k_1, ..., k_n$ be integers such that for any subset $I \subset \{1, 2, ..., n\}$ we have:

$$\left| \bigcup_{i \in I} A_i \right| \geq \sum_{i \in I} k_i$$

Show that there exist sets $B_1, .., B_n$ such that $B_i \subseteq A_i$, and $|B_i| = k_i$ for any $1 \leq i \leq n$, and:

$$B_i \cap B_j = \emptyset \text{ for any } i \neq j$$

Answer: Let $\cup_{i=1}^{n} A_i = \{a_1, a_2, ..., a_m\}$. We construct a graph $G = (V, E)$ such that:

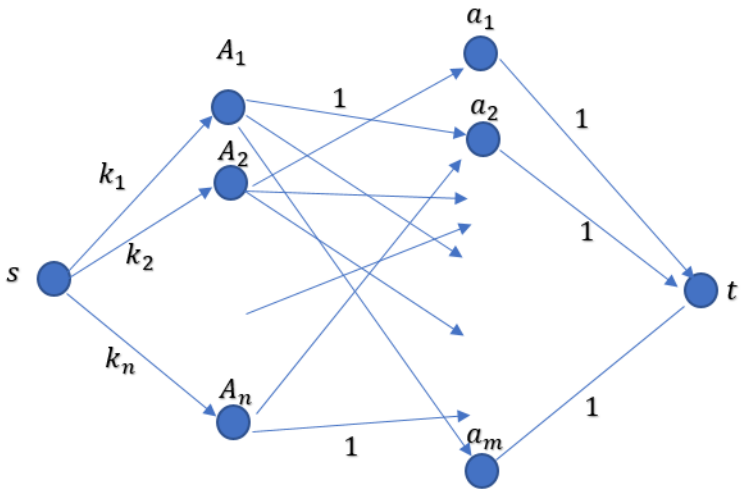
$$V = \{s\} \cup \{A_1, A_2, ..., A_n\} \cup \{a_1, a_2, ..., a_m\} \cup \{t\}$$

$$E = \{(s, A_i) \mid 1 \leq i \leq n\} \cup \{(a_i, t) \mid 1 \leq i \leq m\} \cup \{(A_i, a_j) \mid a_j \in A_i \text{ and } 1 \leq i \leq n, 1 \leq j \leq m\}$$

We define a capacity function by:

$$c(e) = \begin{cases} k_i, & e = (s, A_i); \\ 1, & \text{Otherwise}; \end{cases}$$

The flow network we get is:



We will show that the max flow of this flow network equals $\sum_{i=1}^{n} k_i$. Such a flow clearly defines the required sets $B_1, B_2, ..., B_n$. (For an integer max flow f, define $B_i = \{a_j \mid f(A_i, a_j) = 1\}$).

Let $(S, \overline{S})$ be an s-t cut of the flow network. We have:

$$\text{cap}(S, \overline{S}) = \sum_{A_i \notin S} k_i + \sum_{a_j \in S} 1 + \sum_{A_i \in S \text{ and } a_j \in A_i \text{ and } a_j \notin S} 1$$

For $A_i \in V$ we denote $\Gamma(A_i) = \{a_j \mid (A_i, a_j) \in E\}$, and for a set $\{A_{i_i}, A_{i_2}, ..., A_{i_t}\}$ we denote $\Gamma(\{A_{i_i}, A_{i_2}, ..., A_{i_t}\}) = \cup_{j=1}^{t} \Gamma(A_{i_j})$.
Consider $\Gamma_0 = \Gamma(\{A_1, ..., A_n\} \cap S)$. For any $a_i \in \Gamma_0$, either $a_i \in S$ or $a_i \notin S$. If $a_i \in S$ then a_i contributes 1 to the sum $\sum_{a_j \in S} 1$. If $a_i \notin S$ then a_i contributes 1 to the sum:

$$\sum_{A_i \in S \text{ and } a_j \in A_i \text{ and } a_j \notin S} 1$$

It follows that:

$$\text{cap}(S, \overline{S}) \geq \sum_{A_i \notin S} k_i + \sum_{a_i \in \Gamma_0} 1 = \sum_{A_i \notin S} k_i + \left| \bigcup_{A_i \in S} A_i \right| \geq \sum_{A_i \notin S} k_i + \sum_{A_i \in S} k_i = \sum_{i=1}^{n} k_i$$

We see that for any s-t cut, we have $\text{cap}(S, \overline{S}) \geq \sum_{i=1}^{n} k_i$ so it follows that the maximum flow has value $\sum_{i=1}^{n} k_i$.

4. Let $G = (V, E)$ be an undirected weighted graph with weight function $w : E \to \mathbb{R}_{>0}$. (The weights are positive.)
For a 2 vertices $v_1, v_2 \in V$ we denote:

$$d_2(v_1, v_2) = \min_p \{w(p) \mid p \text{ is a path connecting } v_1 \text{ and } v_2 \text{ with } 2k \text{ edges, } k \in \mathbb{N} \cup \{0\}\}$$

where $w(p) = \sum_{e \in p} w(e)$.

Design an algorithm that gets as input the graph G and a vertex s and calculates $d_2(s, v)$ for all $v \in V$.

Answer: We construct a new graph (undirected) $G' = (V', E')$. The set of vertices V' is a union of two copies of V:

$$V^1 = \{v^1 \mid v \in V\} \quad , \quad V^2 = \{v^2 \mid v \in V\}$$

and

$$V' = V^1 \cup V^2$$

(If one wishes to be formal then $v^1 = (v, 1)$ and $v^2 = (v, 2)$.)
The edges of G' are defined by:

$$E' = \{(u^1, v^2), (u^2, v^1) \mid (u, v) \in E\}$$

and their weights are defined by:

$$w((u^1, v^2)) = w((u^2, v^1)) = w((u, v))$$

Since G' is bipartite, a path from u^1 to v^1 in G' must have an even number of edges. On the other direction, a path from u to v in G with an even number of edges, corresponds to a path from u^1 to v^1 in G'. For example, if $u, v_1, v_2, ..., v_m, v$ is a path with an even number of edges in G, then $u^1, v_1^2, v_2^1, ..., v_m^2, v^1$ is a path from u^1 to v^1 in G'.

It follows that to calculate $d_2(u, v)$ in G, it is enough to calculate the shortest path from u^1 to v^1 in G' (which can be done using Dijkstra's algorithm for example). Indeed any path with an even number of edges from u to v in G, yields a path from u^1 to v^1 in G' with the same weight, and any path from u^1 to v^1 in G' yields a path from u to v in G with even number of edges and the same weight. (Note that we allow paths to use the same edge more then once.)

5. Let G be an undirected connected graph with positive weights on edges. Let T be a minimum spanning tree of G. Show that there exists a way to execute Kruskal's algorithm such that it returns T as a result.

   Answer: An easy way to show that there exists a way to execute Kruskal's algorithm such that it returns T as a result, is the following. We saw that if the edges of G have colors (white or black) then we can construct a MST with maximum number of black edges if we apply Kruskal's algorithm such that whenever there are several edges that can be added to the forest, we chose a black edge. (A proof of the correctness of this algorithm can be found in the solution for quiz 3.)

   It means that if T is some MST of G, and we consider the edges of T as black and all of the other edges of G as white, then T can be found using Kruskal's algorithm, as the (unique) MST with the largest number of black edges.

6. Let L_1 be the language of undirected graphs with Hamiltonian path:

$$L_1 = \{G \mid G \text{ is an undirected graph with a Hamiltonian path}\}$$

Let L_2 be the language of undirected graphs with Hamiltonian cycle

$$L_2 = \{G \mid G \text{ is an undirected graph with a Hamiltonian cycle}\}$$

Prove that $L_2 \leq_p L_1$.

Answer: Given an undirected graph $G = (V, E)$, we construct a new graph G' such that if G has a Hamiltonian cycle then G' has a Hamiltonian path and if G does not have a Hamiltonian cycle then G' does not have a Hamiltonian path.

We construct G' in the following way. We choose a vertex $v_0 \in V$ and we define $G' = (V', E')$ where:

$$V' = V - \{v_0\} \cup \{v_0^1, v_0^2, w^1, w^2\}$$

and:

$$E' = E - \{(v_0, v) \mid (v_0, v) \in E\} \cup \{(v_0^1, v), (v_0^2, v) \mid (v_0, v) \in E\} \cup \{(v_0^1, w^1), (v_0^2, w^2)\}$$

It is easy to see that a Hamiltonian cycle in G defines a hamiltonian path in G'.

On the other hand, if G does not have a Hamiltonian cycle then G' does not have a Hamiltonian path. Indeed, a Hamiltonian path in G' must start in w^1 and end in w^2 (or the other way around - but the graph is undirected). It is easy to see that a Hamiltonian path in G' that starts at w^1 and ends at w^2 corresponds to a Hamiltonian cycle in G.