UNIVERSITY OF TORONTO, DEPARTMENT OF ECE

ECE1762 - Homework 3

Xinyun Lv(1001091178), Yang Wang(1001319227)

October 30, 2015

1 PROBLEM I

solution:
According to the description above, this problem could be mapped to a single source Directed acyclic graph(DAG) as follows:

- Corridors → directed edge
- Elevators → directed edge
- Food Location → vertex
- Exit Point → vertex
- Entry Point → vertex source
- Food Quantity → weight on vertex

So, the input of this problem is a DAG denoted as $G(V,E)$. And the quantity of food at food location could also be described as the weight on every edge directed to the corresponding food location(vertex). The output should be a path P in G that accumulate the max gain(food) across all paths in G from S to any vertex.
For example, suppose we have a DAG as shown in Figure 1.1. We could traverse the vertices in linearized order from left to right since a DAG can always be topologically sorted or linearized. As shown in Figure 1.2, we the linearized version of the DAG shown in Figure 1.1 are then: $\{S,C,A,B,D,E\}$.
The problem equivalent to find the longest path from vertex S to any other vertex. Consider the vertex D in the linearized graph - the only way to get to D from S is through one of its

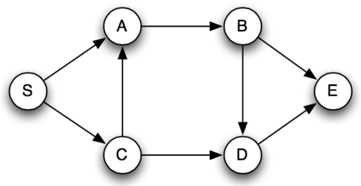

Figure 1.1: A DAG example mapped from problem

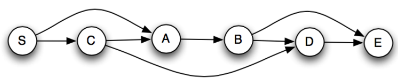

Figure 1.2: Linearized version of G

predecessors: B and C. That means, to compute the longest path from S to D, one must first compute the longest path from S to B (up to the first predecessor), and the longest path from S to C (up to the second predecessor). Once we've computed the longest paths from S to these two predecessors, we can compute the longest path to D by taking the larger of these two, and adding $w(D)$(food quantity in D). If $dist(v)$ is the longest distance from S to vertex v, and $\alpha(v)$ is the actual path, then we can write the following recurrence for $dist(D)$:

$$dist(D) = max\{dist(B) + w(D), dist(C) + w(D)\}$$

Note the subproblems here: $dist(B)$ and $dist(C)$, both of which are âĂIJsmallerâĂİ than $dist(D)$. Similarly, we can write the recurrences for $dist(B)$ and $dist(C)$ in terms of its subproblems:

$$dist(B) = dist(A) + w(B)$$

$$dist(C) = dist(S) + w(A)$$

For the source vertex S, we set:

$$dist(S) = w(S)$$

$$\alpha(S = \{S\}$$

We are now ready to write the recurrence for the longest path from S to any other vertex v in G, which involves first computing the longest paths to all of v's predecessors from S (these are the subproblems).

$$dist(v) = max_{(u,v)\in E} dist(u) + w(v)$$

And, we can compute this bottom-up for each vertex $v \in V$ taken in a linearized order. The final algorithm is shown below:

LONGEST-PATH(G)
Input: Weighted DAG G

Output: Largest path cost in G
Topologically sort G
for each vertex $v \in V$ in linearized order
  do:
  $\text{dist}(v) \leftarrow \max_{(u,v)\in E}\{\text{dist}(u) + w(v)\}$
  $\alpha(v) \leftarrow \max_{(u,v)\in E}\{\alpha(u) \cup v\}$
return $\max_{v\in V}\{\text{dist}(v)\}$

The time complexity is obvious $O(|V| + |E|)$.

2 Problem I

solution:

1. **proof:**
   Without loss of generality, suppose that $I_m > I_t$ are two elements in $\langle I_1, I_2, ..., I_k\rangle$ where $1 \le m \ge k$, $1 \le t \le k$ and $m \ne t$. Suppose that Greedy-Schedule determines order $I_m \succ I_t$ on I. Greedy-Schedule determines an order $I_m \prec I_t$ on I'.

   From definition of order restriction, for $I_m, I_t \in I$ and $I_m, I_t \in I'$. I_m precedes I_t **iff** it also determines an order on I' where I_m precedes I_t yielding a contradiction to the assumption that $I_m \succ I_t$ on I, $I_m \prec I_t$ on I'. The opposite direction is the same. Thus, Greedy-Schedule determines an order $I_1 \succ I_2 \succ ... \succ I_k$ on I if Greedy-Schedule determines an order $I_1 \succ I_2 \succ ... \succ I_k$ on I'.

2. **proof:**
   we first prove if (a), (b) and (c) happens $\Rightarrow$ Greedy-Schedule is not correct.

   If we use algorithm First Starting Time First which means we choose interval in I with the earliest start time. Let interval $I[a], I[b], I[c] \in I$ where $s_a < s_b < s_c$ and $f_c < f_a$ and $f_b < s_c$. For these three intervals, they meet (a), (b), (c) as shown in the figure.

   If we choose interval in I with the earliest start time, then we will choose I_a instead of I_b and I_c. For optimal schedule, we should choose I_b and I_c instead of I_a since we could schedule more intervals within the same time slot. Thus, Greedy-Schedule is not optimal.

   Greedy-Schedule is not correct $\Rightarrow$ (a), (b), (c) happens.

   If Greedy-Schedule is not optimal, which means it will choose less intervals compare with the optimal set. If Greedy-Schedule GS is not optimal and $I_a, I_b, I_c \in I$. As GS is not optimal we can assume that GS picked I_a instead of I_b, I_c. But for the optimal schedule set(OS), it choose I_b and I_c instead of I_a. Which means that I_a should overlap with I_b and I_c. So they could not be chosen at the same time. Which meets the situation (a). As I_b and I_c could be chosen together which implies that I_b and I_c does not overlap with each other. So it also meets situation (b). As in GS, I_a is chosen first, so we could not choose I_b and I_c. Which implies that on input I Greedy-Schedule determines an ordering O where $I[a]$ comes before $I[b]$ and $I[c]$, so situation(c) also happens.

3. **proof:**
According to the conclusion in question (2), we know that Greedy-Schedule is correct iff (a), (b), (c) not happens. So to prove LSTF is correct, is equivalent to prove (a), (b) and (c) cannot happen.

Suppose that for LSTF, (a), (b), (c) happens. For $I_a, I_b, I_c \in I$. We let s_i, f_i be the start time and finish time of interval I_i. If (a) is true, here without loss of generality, we let I_b precedes I_c, then $s_a < f_b$ (1) and $s_c < f_a$. If (b) is true. $f_b < s_c$ (2). From (1) and (2) we know that $s_a < f_b < s_c$ (3). If (c) is true, according to the rules of LSTF, as I_a precedes I_b, I_c, $s_a > s_b > s_c$ which contradict (3). So, for LSTF (a), (b), (c) cannot happen together. Thus LSTF is correct.

3 PROBLEM III

1. **proof:**
Since for every $S \subseteq U, \mid S \cap U_i \mid \geq 2$, functions $h_1, ..., h_i$ are not ferfect for S. Then we must have $\mid U_t \mid \leq 1$.

2. **proof:**
Since $U_0 = U$ and we let $U_{i+1} = U_i \cap h_{i+1}^{-1}(\alpha)$ for $i < t$, where α is such that $\mid U_i \cap h_{i+1}^{-1}(\alpha) \mid \geq \mid U_i \cap h_{i+1}^{-1}(j) \mid$ for every $j \in [0, ..., m-1]$. Then $\mid U_{i+1} \mid \geq \frac{|U_i|}{m}$ and hence $\mid U_{i+1} \mid \geq \frac{N}{m^{i+1}}$. Also functions $h_1, ..., h_{i+1}$ are constant on U_{i+1} and hence $U_t \leq 1$. Thus $1 \geq \frac{N}{m^t}$ or $t \geq \frac{logN}{logm}$. Which implies
$$\mid H \mid \geq \frac{logN}{logm}$$

4 PROBLEM IV

5 PROBLEM V