CS624: Analysis of Algorithms

Final Exam – Practice

Instructor – Nurit Haspel

1. **Greedy algorithms: Greedy Algorithms: (30%)**

   You drive from Boston to Washington DC. Your gas tank, when full, holds enough gas to travel m miles and you have a map that gives you the distances between gas stations on the way. Let $d_1 < d_2 < ... < d_n$ be the locations of the gas stations, where d_i is the distance from Boston to the gas station (the list is sorted). Assume that the distance between two consecutive gas stations is at most m miles.

   Your goal is to make as few stops for gas as possible along the way without getting stuck. A greedy algorithm to solve the problem is to always drive as far as possible before stopping for gas. More formally, let c_i be the destination with distance d_i from Boston:

   $S = \emptyset$

   $last = 0$

   for $(i = 1..n)$

   if $(d_i - last > m)$ // d_i is the distance of first station that's too far

   $S = S \cup c_{i-1}$ // Add the one before (the last that's not too far)

   $last = d_{i-1}$

   return S

   (a) Show the problem has the optimal substructure property. Start by: Let's say S is an **optimal** solution. Given that we stop for gas at stop at d_i, then what can you say about the the parts of the road uncovered by d_i?

   The parts of the road uncovered by d_i must be optimal as well (both m miles before and m miles after d_i), since otherwise we could cut them out and replace them by a better solution.

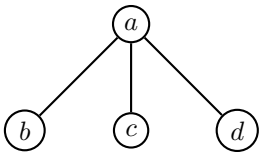
   (b) Show that the greedy choice is safe. Specifically, let g be the first stop selected by the greedy algorithm (the farthest from Boston you can stop without getting stuck). Show that there is an optimal solution S that contains g as its first stop. Explain your answer.

   The answer is similar to the activity scheduling problem we saw in class. Say we have an optimal solution S. If S contains g – the farthest from Boston you can drive without stopping for gas, we're ok. Otherwise, let's look at f, the first gas stop on the way. We can take f out and replace it by g. Since g is farther than f, it's not going to spoil S (we are not going to get stuck before the next stop, since g is closer to it, and we're not going to get stuck before g, since g is still within the distance). Therefore, the new solution is still optimal.

2. (20%) **Graphs:** In this question we refer to a node as "undiscovered" if it has not been seen yet by a graph walk (colored white in class). A node is "discovered" if it has been seen but not yet done (colored gray in class). A node is "processed" if it is done (colored black in class).

   (a) (10%) Show an example of a graph $G = (V, E)$ such that a BFS walk yields $O(|V|)$ processed (gray) nodes at some given moment. Explain. **To get the full mark your example should contain at least four nodes.**

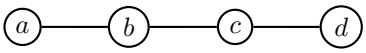
   Answer: See figure:



When the BFS starts in a. When a is done, all the other vertices are gray.

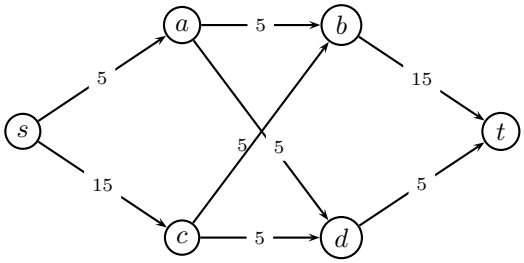
(b) (10%) Show an example of a graph $G = (V, E)$ such that a DFS walk yields $O(|V|)$ processed (gray) nodes at some given moment. Explain. **To get the full mark your example should contain at least four nodes.**

Answer: See figure:



When the BFS starts in a. When d is explored, all vertices are gray.

3. (20%) **Flow:** Given the following flow graph:



Show the maximum flow for this graph. No need to show the residual graphs. Write the augmenting paths you find, the value of the flow for each path and the value of the overall maximum flow.

Answer:

The paths are:

- $s \to a \to b \to t$ (5)
- $s \to c \to d \to t$ (5)
- $s \to c \to b \to t$ (5)

The overall flow is 15.

4. (25%) **NP:** The directed Hamiltonian Path (HAM-Path) is the problem of finding a simple path that goes through every vertex in a directed graph once. To show that the problem is NP-complete we use a reduction from the directed Hamiltonian cycle problem (HAM-Cycle) – we start from an instance $G = (V, E)$ to the HAM-cycle problem. We create a directed graph $H = (V', E')$ as follows: Select an arbitrary node $u \in V$ and split it to two nodes, u_{in} and u_{out}. Every edge (u, v) in G will now become (u_{out}, v) in H and every edge (v, u) becomes (v, u_{in}) in H. The other vertices and edges remain unchanged.

(a) (6%) Show that directed HAM-Path is in NP.

Answer: This one is easy. Given the vertices in order, all we have to do is verify they constitute a simple path (no cycles) and that they are all the vertices in the graph. This takes linear time in the number of vertices.

(b) (7%) Show that the reduction described above is polynomial.

Answer: All we have to do is to add at most $|V| - 1$ edges from u_{out} to all the rest, and at most $|V| - 1$ edges from all the rest to u_{in} (depending on the degree of u). This is linear in the number of vertices, and we add a linear number of edges and one vertex, so the space is polynomial.

(c) (7%) Show that the graph G has a HAM-Cycle iff H has a HAM-Path. Don't forget the two directions.

Answer:

$\Rightarrow$ If G has a HAM-Cycle $\{v_1 = u, v_2, \ldots, v_n, v_1 = u\}$ (since it's a cycle it doesn't matter where we start the cycle, so we start with u). This corresponds to the path $\{u_{out}, v_2, \ldots, v_n, u_{in}\}$. Since there is an edge (u, v_2) in the original graph, there is an edge (u_{out}, v_2) in H, and since there is an edge (v_n, u) in the original graph, there is an edge (v_n, u_{out}) in H.

$\Leftarrow$ If H has a HAM-PATH, it must start with u_{out} and end with u_{in}, since they don't have incoming and outgoing edges, respectively. But they both represent the same node u in G, so this path corresponds to a cycle in G.