Analysis of Algorithms Practice Session – III

(2010-2011, Spring Term)

G. Selda UYANIK

(seldauyanik@itu.edu.tr)

Q1b: (5 points)

- What is the strong connectivity for a graph?
- For a given graph G, how could we test if it is strongly connected or not? Explain step by step. You may refer to some algorithms discussed in the class without writing them.

Q1c: (5 points)

- The head of a CE department wants to prepare a prerequisite graph of the courses offered in the department. What kind of structure s/he form? Explain.
- Considering the structure given in Q1c, how can one list all the courses that are prerequisites of a given course?

Q2a: (18 points)

- Write an algorithm that finds **all** two-edge long paths with **distinct** nodes in a directed graph G =(V,E)

Q2c: (7 points)

- If n = |V| and m = |E|, what is the complexity of an efficient algorithm solving Q2a for linked list and matrix representation of the graph in the theta notation? Explain.

Q4: (10 points)

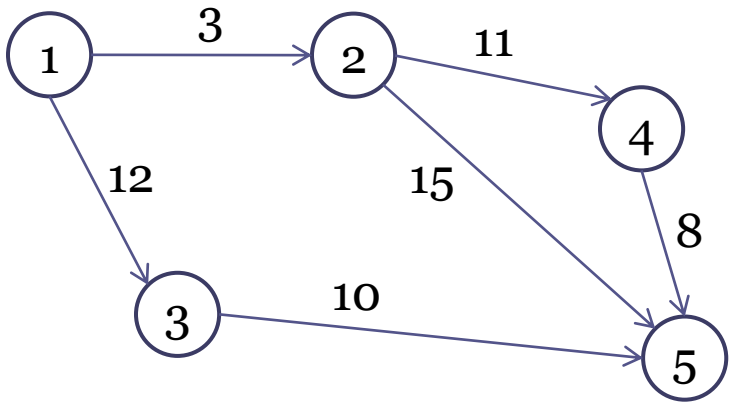
- Consider the problem of scheduling two jobs *a* and *b* wich take an hour each.
- Asume that each job can start either at time t=0 or t=1.
- Let p_0(a) and p_1(a) show the probability that job a starts at time 0 and 1 respectively. Similarly, p_0(b) and p_1(b) show the probability that job b starts at time 0 and 1 respectively.

- Compute the expected number of jobs that can be scheduled when:
  - *i.* $p_0(a) = p_1(a) = p_0(b) = p_1(b) = 0.5$ (each job appears uniformly random)
  - *ii.* $p_0(a) = p_0(b) = 0.7$, $p_1(a) = p_1(b) = 0.3$ (jobs are more likely to appear earlier)

Hint: In order to find the expected # of jobs, for each possible scenario of jobs, you'll need to compute the probability of that scenario and multiply it with # of jobs that can be scheduled for that scenario.

Q5a: (10 points)

- Consider Dijkstra's shortest path algorithm to find shortest path between two nodes s and t in a directed graph. Each edge is represented with the two nodes u and v it connects: e (u,v). The legth of an edge is given by $l_e \geq 0$.
- Give pseudo code of algorithm and show its execution to find shortest s = 1 and t=5 path on graph below:



Dijkstra's Algorithm

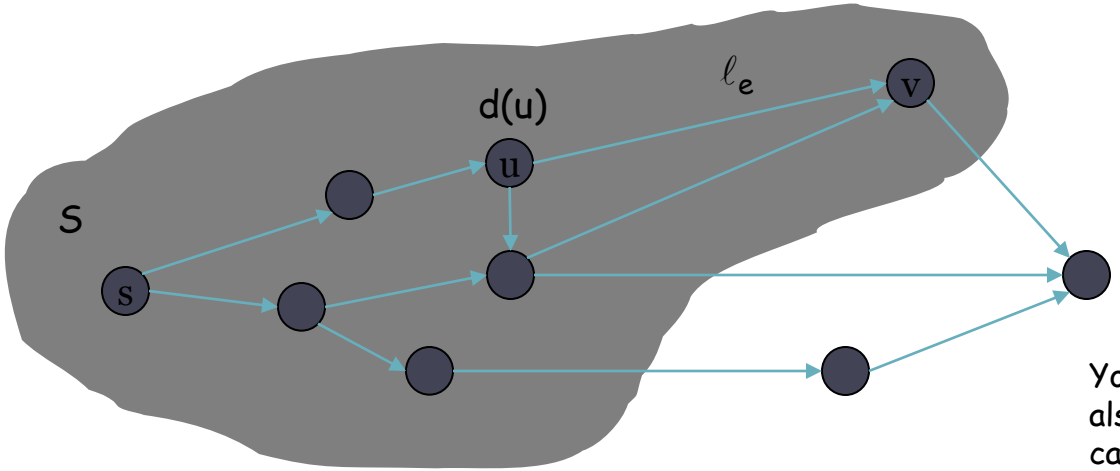
Dijkstra's algorithm.

- Maintain a set of explored nodes S for which we have determined the shortest path distance d(u) from s to u.
- Initialize S = { s }, d(s) = 0.
- Repeatedly choose unexplored node v which minimizes

$$\pi(v) = \min_{e=(u,v):\, u \in S} d(u) + \ell_e,$$

add v to S, and set d(v) = π(v).

shortest path to some u in explored part, followed by a single edge (u, v)



You keep d(u) for each noe u, if you also keep how you reached that u, you can also find shortest path nodes, not just its length …

Algorithm-Dijkstra

```
function Dijkstra ( L[1..n, 1..n] ): array [2..n]
//Finds the lenght of the shortest path from node1 to each of the other
nodes of the graph with n nodes
//Input: L(i,j): Length of the edge between vertices i and j

array D[2..n]
{initialization}
C <- {2, 3, 4, ..., n}                                  : unexplored nodes!
S <- {1}
for i <-2 to n do
        D[i] <- L[1, i]
 repeat (n-2)  times
        v <- some element of C minimizing D[v]
        C <- C\{v}
        S <- S U {v}
        for each (w member-of C) do
                D[w] <- min (D[w], D[v]+L[v,w])
endrepeat
return D
```

$\Theta(n^2)$

« An algorithm is greedy if it builds a solution in small steps, choosing a decision at each step myopically [=locally, not considering what may happen ahead] to optimize some underlying criterion. »

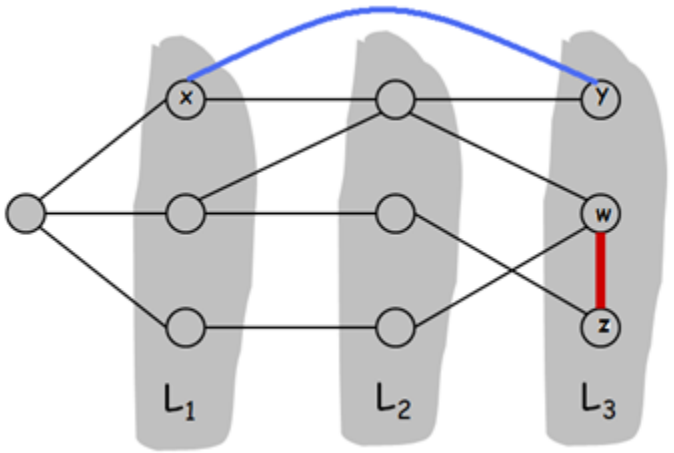
Q2a: (10 points)

- In the lecture we proved the following lemma:
- ***Lemma:*** *Let G be a connected graph, and let L_0, …, L_k be the layers produced by BFS starting at node s. Exactly one of the following holds.*

(i) No edge of G joins two nodes of the same layer, and G is bipartite.

(ii) An edge of G joins two nodes of the same layer, and G contains an odd-length cycle (and hence is not bipartite).

- Part (ii) can be depicted by edge (w,z). In the following BFS layers is edge (x,y) possible? Why or why not?



Q3b: (7 points)

Q3: Give, using "big oh" notation, the worst case running times of the following procedures as a function of n. Make it as precise as possible and show all your work.

```
procedure mystery1 ( n: integer);
        var
             i, j, k: integer;
        begin
            for i:= 1 to n-1 do
            for j:= i + 1 to n do
                for k := 1 to j do
                            { some statement requiring O(1) time }
        end
```

Q3c: (7 points)

Q3: Give, using "big oh" notation, the worst case running times of the following procedures as a function of n. Make it as precise as possible and show all your work.

```
procedure veryodd ( n: integer );
    var
        i, j, x, y: integer;
    begin
       for i := 1 to n do
           if odd(i) then begin
              for j := i to n do
                     x := x + 1;
              for j := 1 to i do
                     y := y + 1
           end
    end
```

Q3d: (7 points)

Q3: Give, using "big oh" notation, the worst case running times of the following procedures as a function of n. Make it as precise as possible and show all your work.

```
procedure mystery2 ( n: integer );    /assuming n is a positive power of 2/

        var
            x, count: integer;
          begin
            count := 0;
            x := 2;

            while x < n do begin
                x := 2 * x;
                count := count + 1
            end;

            writeln(count)
          end
```

http://www.informatik.uni-ulm.de/acm/Locals/2009/html/dark.html

Problem: Dark roads

- Economic times these days are tough, even in Byteland. To reduce the operating costs, the government of Byteland has decided to optimize the road lighting. Till now every road was illuminated all night long, which costs 1 Bytelandian Dollar per meter and day.
- To save money, they decided to no longer illuminate every road, but to switch off the road lighting of some streets. To make sure that the inhabitants of Byteland still feel safe, they want to optimize the lighting in such a way, that after darkening some streets at night, there will still be at least one illuminated path from every junction in Byteland to every other junction.

- What is the maximum daily amount of money the government of Byteland can save, without making their inhabitants feel unsafe?

Input Specification
two numbers ***m*** and ***n***, the number of junctions in Byteland and the number of roads
Then follow ***n*** integer triples ***x, y, z*** specifying that there will be a bidirectional road between ***x*** and ***y*** with length ***z*** meters

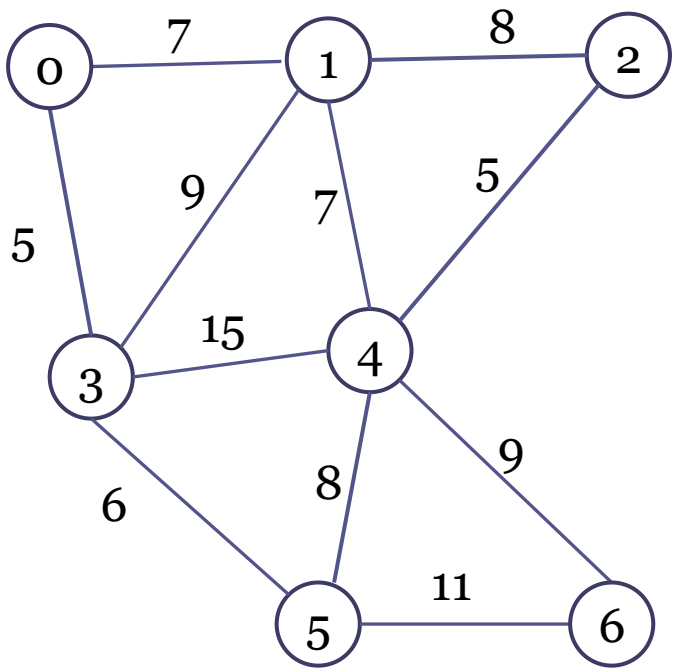
Output Specification
For each test case print one line containing the maximum daily amount the government can save.

http://www.informatik.uni-ulm.de/acm/Locals/2009/html/dark.html

Problem: Dark roads

- Sample Input

```
7 11
0 1 7
0 3 5
1 2 8
1 3 9
1 4 7
2 4 5
3 4 15
3 5 6
4 5 8
4 6 9
5 6 11
```



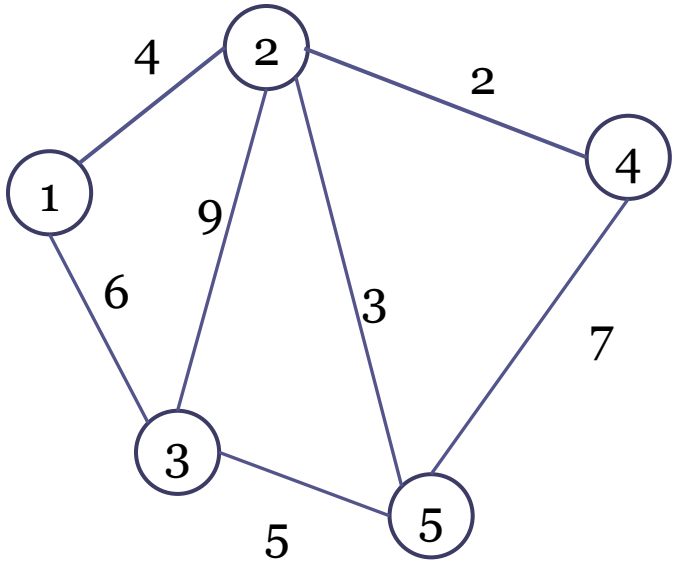
- Sample Output

```
51
```

if |V| <= 200000 and |E| <= 200000, which algo to choose?

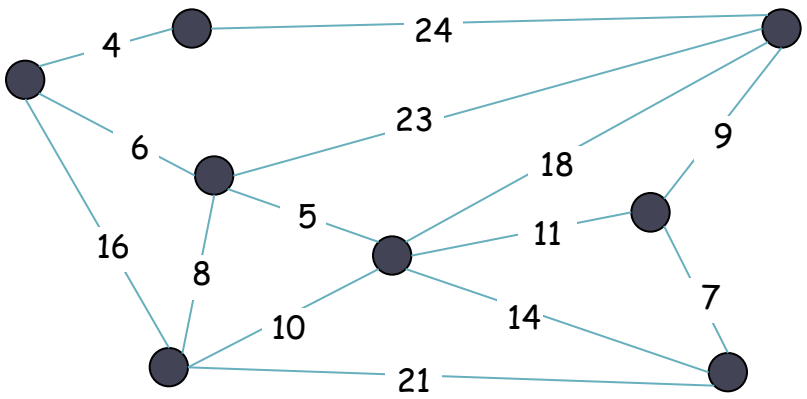
Q1a: (10 points)

- Compute Minimum Spanning Tree (MST) for following graph using Prim's algorithm. Start your algorithm with node 1 and provide all details of your steps.

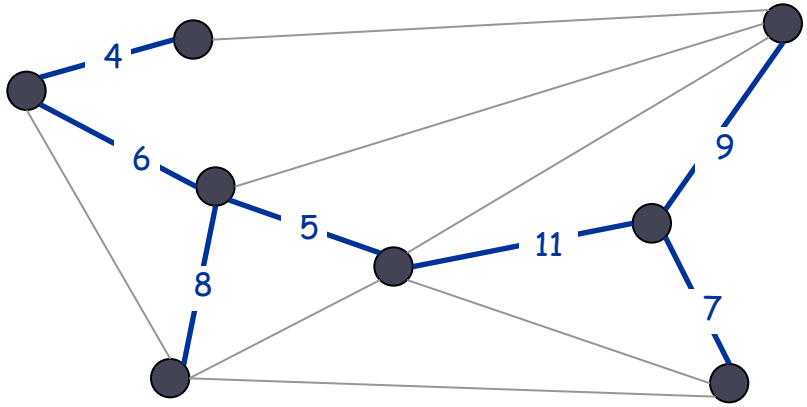


Minimum Spanning Tree

Minimum spanning tree. Given a connected graph G = (V, E) with real-valued edge weights c_e, an MST is a subset of the edges $T \subseteq E$ such that T is a spanning tree whose sum of edge weights is minimized.



G = (V, E)



T, $\Sigma_{e \in T} c_e = 50$

Cayley's Theorem. There are n^{n-2} spanning trees of K_n.

↑

can't solve by brute force

Greedy Algorithms for MST

- Kruskal's algorithm: Start with $T = \phi$. Consider edges in ascending order of cost. Insert edge e in T unless doing so would create a cycle.

- Prim's algorithm: Start with some root node s and greedily grow a tree T from s outward. At each step, add the cheapest edge e to T that has exactly one endpoint in T.

- Reverse-Delete algorithm: Start with T = E. Consider edges in descending order of cost. Delete edge e from T unless doing so would disconnect T.

Q1b: (10 points)

- Consider a graph with unequal and positive edge costs. Assume you run Prim's algorithm starting with a node s and find MST T. Assume you run Prim's algorithm again starting with another s' ≠ s and you fing MST T'.

- Is T = T'? Proove or disprove.