# CSE317 Design & Analysis of Algorithms

Final Examination – Spring'16

Max Marks: 70 Time Allowed: 3 hours

Answer all questions. Use seperate answer sheet. Please give clear and rigorous answers. Be to the point. Show your work.

**Name:** _____ **ERP:** _____

**Question 1: Graphs** ..... *10 marks*

(a) [5 marks] Consider the following problem. You are given a connected graph $G$ with $n$ nodes and $m$ edges. You want to remove as many edges as possible and still keep the graph connected.

i. How many edges must remain in the graph in order to ensure that $G$ is connected?

ii. Give an efficient algorithm to solve the problem

(b) [5 marks] Suppose the degree requirements for a computer science major are organized as a DAG (directed acyclic graph), where vertices are required courses and an edge $(x, y)$ means course $x$ must be completed prior to beginning course $y$. Make the following assumptions:

- All prerequisites must be obeyed.
- There is a course, CS101, that must be taken before any other course.
- Every course is offered every semester.
- There is no limit to the number of courses you can take in one semester.

Describe an efficient algorithm to compute the minimum number of semesters required to complete the degree and analyze its running time. Hint: Doable in linear time.

**Question 2: Analysis** ..... *8 marks*

(a) [4 marks] In each of the following situations, indicate whether $f = O(g)$, $f = \Omega(g)$, or both.

| | $f(n)$ | $g(n)$ |
|---|---|---|
| (i) | $n - 100$ | $n - 200$ |
| (ii) | $10n$ | $\log(n^2)$ |
| (iii) | $2^n$ | $2^{n+1}$ |
| (iv) | $n!$ | $2^n$ |

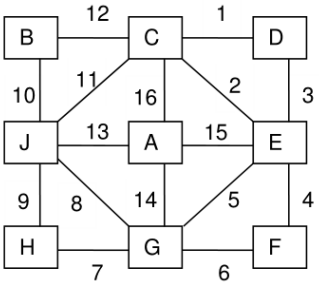
(b) [2 marks] Consider the recurrence $T(n) = 2^{\log T(n/2)} + T(n/2)$, where $T(1) = 1$. Use induction to prove that $T(n) = O(n)$. Dont forget to write down precisely what you are trying to prove and to include the base case, inductive hypothesis and inductive step.

(c) [1 mark] What is $\sum_{i=1}^{n} i$?

(d) [1 mark] What is the solution of the recurrence $T(n) = T(n-1) + \log n$?

**Question 3: Graphs** ..... *12 marks*

(a) [6 marks] Clearly indicate the following trees in the weighted graph pictured below. Some of these subproblems have more than one correct answer. (You do not need to show the intermediate steps, just give the required trees)

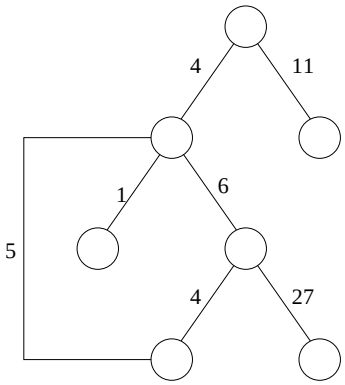



i. A BFS tree rooted at $A$

ii. A shortest-path tree rooted at $A$

iii. A minimum spanning tree

(b) Consider an undirected weighted graph which is formed by taking a binary tree and adding an edge from *exactly one* of the leaves to another node in the tree. We call such a graph a *loop-tree*. An example of a loop-tree could be the following:



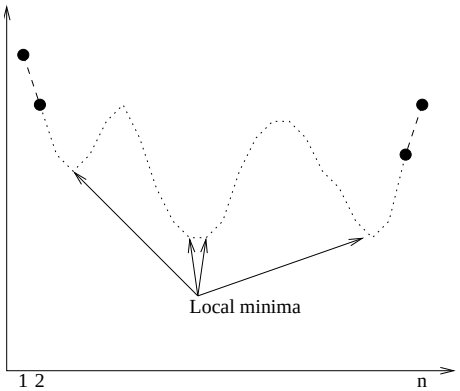

Let $n$ be the number of vertices in a loop-tree and assume that the graph is given in the normal edge-list representation without any extra information. In particular, the representation does not contain information about which vertex is the root.

i. [2 marks] How long time would it take Prim's or Kruskal's algorithms in worst case to find the minimal spanning tree of a loop-tree? Make your bound as tight as possible.

ii. [4 marks] Describe and analyze a more efficient algorithm for finding the minimal spanning tree of a loop-tree. Remember to prove that the algorithm is correct.

**Question 4: Divide & Conquer** ..... *8 marks*

(a) [5 marks] In this problem we consider an array $A$ of length $n$ for which we know that $A[1] \geq A[2]$ and $A[n-1] \leq A[n]$. We say that $A[x]$ is a *local minimum* if $A[x-1] \geq A[x]$ and $A[x] \leq A[x+1]$. Note that $A$ must have at least one local minimum.




We can obviously find a local minimum in $O(n)$ time by scanning through $A$. Describe an $O(\log n)$ algorithm for finding a local minimum.

(b) [3 marks] Solve the following recurrence: $T(n) = 2T(n/2) + n^2$ (without using master theorem)

**Question 5: Dynamic Programming** ..... *13 marks*

(a) *Knapsack Problem.* You have four types of items, where the $i$th item type has an integer weight $w_i$ and a real value $v_i$.

| Item | Weight ($w_i$) | Value ($v_i$) |
|---|---|---|
| 1 | 1 | 3 |
| 2 | 3 | 4 |
| 3 | 6 | 8 |
| 4 | 4 | 6 |

i. [2 marks] Write the recurrence for computing the subset of items of maximum value such that the total weight of selected items is at most the capacity $W$ of the knapsack.

ii. [4 marks] You are trying to fill a knapsack of total capacity $W = 6$ with a selection of items of maximum value. Use the recurrence from the above part and fill in the appropriate table.

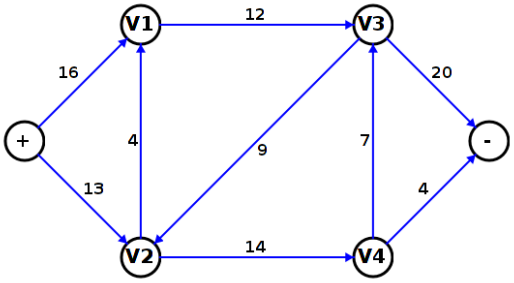
(b) In the PARTITION problem, you are given a list of $n$ integers, $x_1, x_2, \ldots, x_n$ all in the range 1 to $k$. You want to determine if there is some subset of the integers that sums to exactly $S/2$, where $S$ is the sum of all the integers in the list.

i. [5 marks] Give an efficient algorithm to solve this problem and analyze your algorithm. Hint: Let $f(i, y) = 1$ if there is some subset of the first $i$ integers that sums exactly to $y$ and let it be 0 otherwise.

ii. [2 marks] The PARTITION problem is NP-Complete. Explain how this fact reconciles with your ability to devise an efficient algorithm above. Hint: This answer is no more than two sentences.

**Question 6: Network Flows** ..... *10 marks*

(a) Consider the following network (the numbers are edge capacities, + is source, and - is sink).




i. [2 marks] List two different $s$-$t$ cuts along with their capacities.

ii. [6 marks] Find the maximum $s$-$t$ flow and a minimum cut using Ford-Fulkerson algorithm.

(b) [2 marks] Suppose someone presents you with a solution to a max-flow problem on some network. Give a linear time algorithm to determine whether the solution does indeed give a maximum flow.

**Question 7: Reductions and NP-completeness** ..... *9 marks*

(a) [2 marks] What is an independent set in graph $G = (V, E)$? Give a rigorous definition and an example.

(b) [1 mark] If the minimum vertex cover of a graph $G$ is of size $x$, what is the size of the maximum independent set? Assume that G has $n$ nodes and $m$ edges.

(c) [6 marks] For each of the following statements, answer if it is true or false. Briefly justify your answer.

i. It is known that there can be no polynomial-time algorithm for a NP-complete problem.

ii. For a problem $X$, if $X \in \mathsf{P}$, then $X \in \mathsf{EXP}$.

iii. A problem cannot be both in P and NP.

iv. If $A$ and $B$ are problems that belong to NP-complete, then $B \leq_P A$?