Faculty of Science
Final Examination

Computer Science COMP-251A
Data Structures and Algorithms

Examiner: Prof. Claude Crépeau **Date:** Dec. 22, 2015

Associate Examiner: Prof. Prakash Panangaden **Time:** 14:00 – 17:00

INSTRUCTIONS:
- This examination is worth 50% of your final grade.
- The total of all questions is 100 points.
- Each question is assigned a value found in brackets next to it.

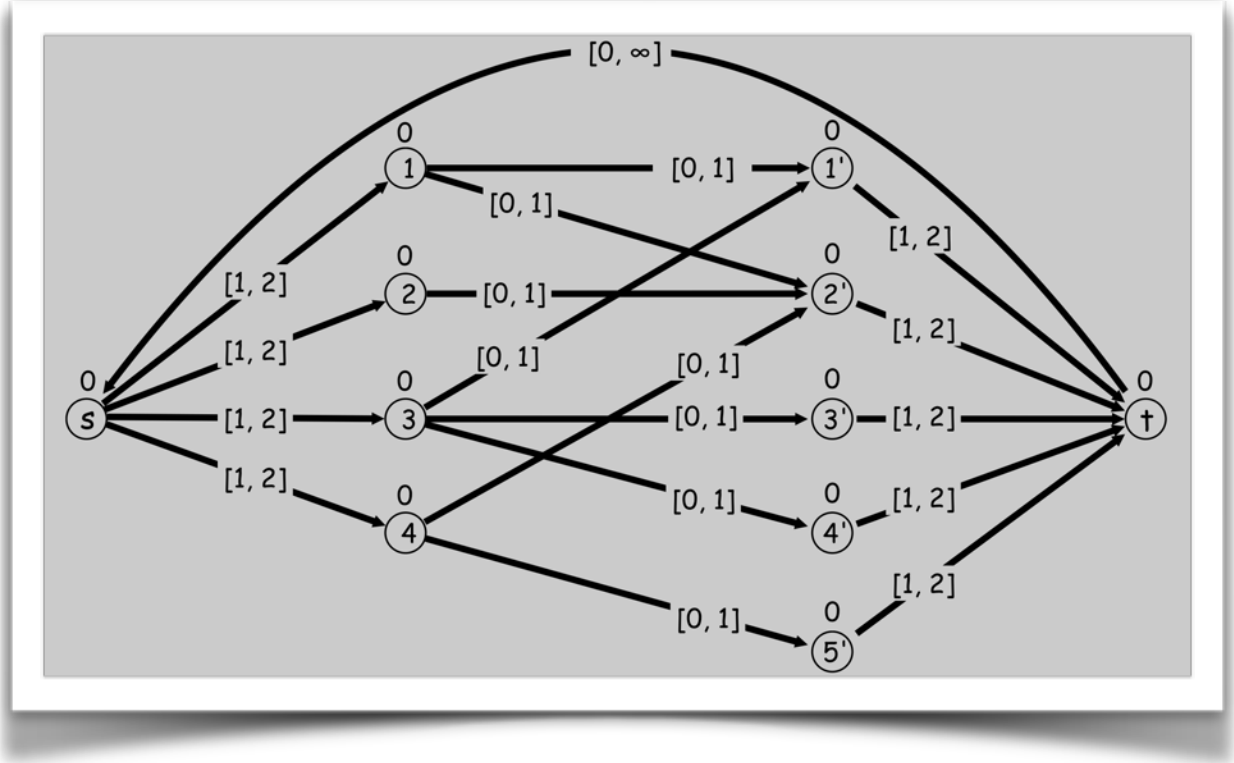
- O P E N • B O O K S •/• O P E N • N O T E S

- Faculty standard calculator permitted only.
- This examination consists of 5 pages including title page.
- This examination consists of 5 questions.

SUGGESTION : read all the questions and their values *before* you start.

[20%] 1) Circulation

Consider the following Circulation with Demands and Lower Bounds problem instance:



A) Write a Survey Design problem instance that would have this representation according to our method of transformation.

B) Remove the lower bounds from the above problem instance to create an equivalent Circulation with Demands (without lower bounds) problem instance.

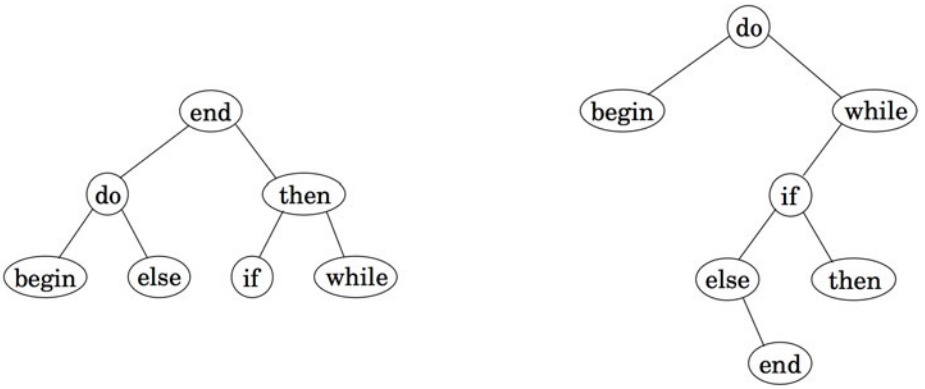
C) Explain how you would solve the above problem. Tell me what algorithm you would use and what the running time would be on a general instance.

[20%] 2) Optimal BST.

Suppose we know the frequency with which keywords occur in programs of a certain language, for instance:

begin	5%
do	40%
else	8%
end	4%
if	10%
then	10%
while	23%

Figure 6.12 Two binary search trees for the keywords of a programming language.



We want to organize them in a binary search tree, so that the keyword in the root is alphabetically bigger than all the keywords in the left subtree and smaller than all the keywords in the right subtree (and this holds for all nodes). Figure 6.12 has a nicely-balanced example on the left. In this case, when a keyword is being looked up, the number of comparisons needed is at most three: for instance, in finding "**while**", only the three nodes "**end**", "**then**", and "**while**" get examined. But since we know the frequency with which keywords are accessed, we can use an even more fine-tuned cost function, the *average number of comparisons* to look up a word. For the search tree on the left, it is

cost = 1 x (4%) + 2 x (40% + 10%) + 3 x (5% + 8% + 10% + 23%) = 2.42 comparisons.

By this measure, the best search tree is the one on the right, which has a cost of 2.18 comparisons.

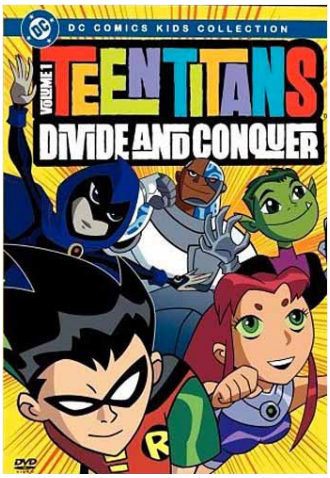
Give an efficient (dynamic programming) algorithm for the following task.

Input: n words (in sorted alphabetical order); frequencies of these words: $p_1, p_2, \ldots, p_n$.
Output: The binary search tree of lowest cost (defined above as the expected number of comparisons in looking up a word).

HINT: Consider how you could find the optimal BST by selecting the root and combine it in a BST that has a left optimal BST for the words previous to the root in alphabetical order and a right optimal BST for the words following the root in alphabetical order.

[20%]

3) Euclid.



Consider the following (variation of) Euclid's algorithm for finding the Greatest Common Divisor of two integers:

function Euclid(a, b)
Input: Two integers a and b with $a \geq b \geq 0$
Output: gcd(a, b)

if $b == 0$: **return** a
$c = a \bmod b$
if $c == 0$: **return** b
return Euclid(c, $b \bmod c$)

A) Show that if "$a \geq b > 0$ and $c > 0$" then "$c < a$ and $(b \bmod c) < b/2$".

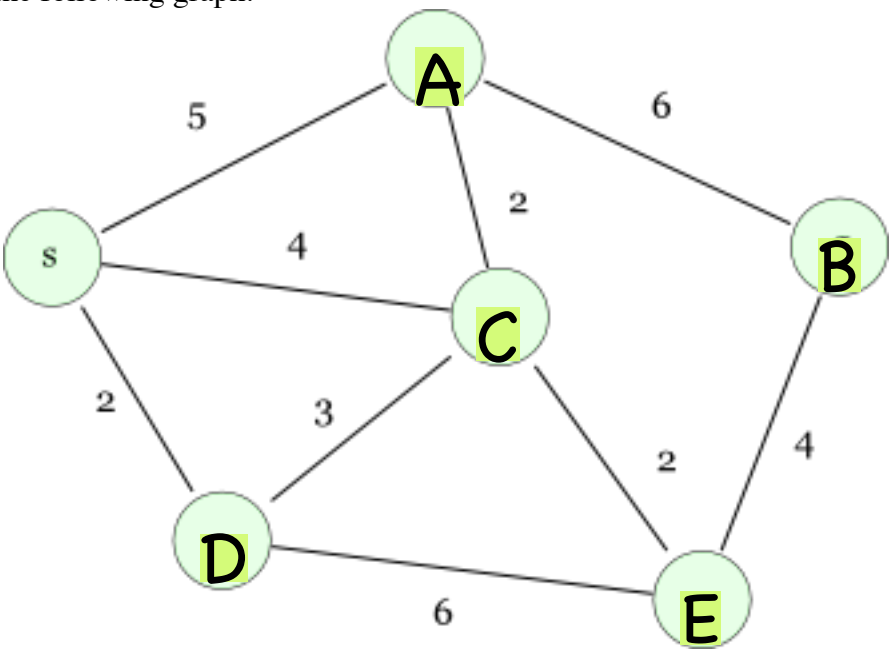
B) Explain why this algorithm may be considered as a divide-and-conquer algorithm.

C) Write the recurrence for the worst-case running-time of this algorithm and solve it. (Assume all the arithmetic operations involved have constant $O(1)$ cost, and let n, the input size, be an upper bound $n \geq |a| \geq |b|$ on the number of bits of both inputs.)

[20%]

4) Shortest Paths.

Consider the following graph:

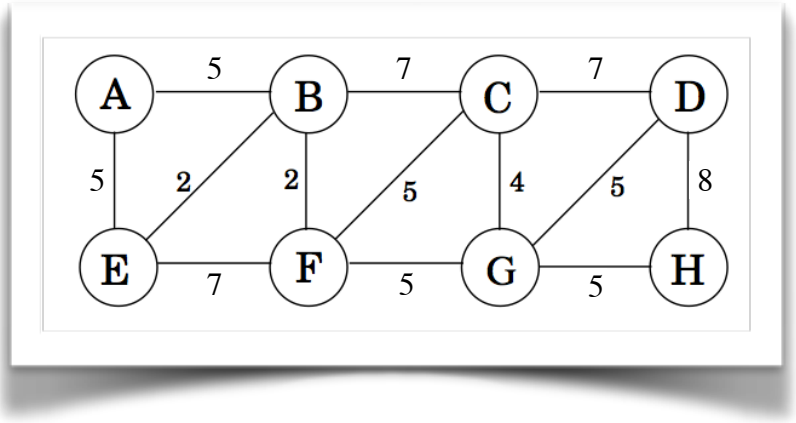


Find the shortest paths from the source s to every other vertex {A,B,C,D,E} in the graph. Choose the algorithm you wish to use, and show the step by step progression of your algorithm on this instance.

[20%]

5) Minimum Spanning Tree.

Consider the following graph:



(a) What is the cost of its minimum spanning tree ?

(b) How many minimum spanning trees does it have ?

(c) Suppose Kruskal's algorithm is ran on this graph. In what order are the edges added to the MST ? For each edge in this sequence, give a cut that justifies its addition.