EC 504
Fall, 2018
Exam 3

Monday, December 17, 2018

Instructions

- Put your name on the first page.
- This exam is open notes, but no consultation with anyone other than the instructors are permitted during the exam.
- You have one hour and 45 minutes to complete this exam. There are 7 questions. All work must be shown on the paper to be counted.

1. (20 pts) Answer True or False to each of the questions below, **AND YOU MUST PROVIDE SOME FORM OF EXPLANATION (very brief is OK)**. Each question is worth 2 points. Answer true only if it is true as stated, with no additional assumptions.

   (a) Consider a minimum spanning tree T in a connected, weighted undirected graph (V, E) where the weights have different values. Then, for every node i, the minimum spanning tree must contain the arc $\{i, j\}$ with the smallest weight connected to node i.

   (b) Dynamic programming can be used to solve the integer knapsack problem in pseudo-polynomial time.

   (c) Assume that there is a decision problem which can be described in an input with n bits. Assume that the worst-case instance of this decision problem of size n bits requires $O(2^n)$ operations to solve. Then, the decision problem belongs to class NP.

   (d) Finding the minimum-distance tour in a traveling salesperson problem is an NP-Hard problem.

   (e) The problem of determining whether a graph has an Eulerian cycle is in class P.

   (f) Suppose we have an R-tree of order M, with minimum keys m, where the R-tree stores data that is indexed by points in 2-d space. If the R-tree is of height (Root level plus next level plus leaf level), then the maximum number of data entries stored in the R-tree is M^2.

   (g) Suppose we have n keys stored in an R-tree. Then, the worst-case time to find a key is $O(\log(n))$.

   (h) Given any instance of a traveling salesperson problem, we can find an approximate algorithm that computes a tour of all the nodes with distance no worse than 2 times the distance of the optimal tour.

2. (10 pts) Consider the following 2-dimensional set of points:

   $$(4, 8), (3, 4), (2, 9), (1, 2), (2, 3), (4, 4), (6, 7), (7, 9), (1, 10)$$

   Arrange the elements in a PR quadtree with value range 0-10 for both the first and second coordinates.

3. (10 pts) Consider the following 2-dimensional set of points:

   $$(4, 8), (3, 4), (2, 9), (1, 2), (2, 3), (4, 4), (6, 8), (7, 9), (1, 10)$$

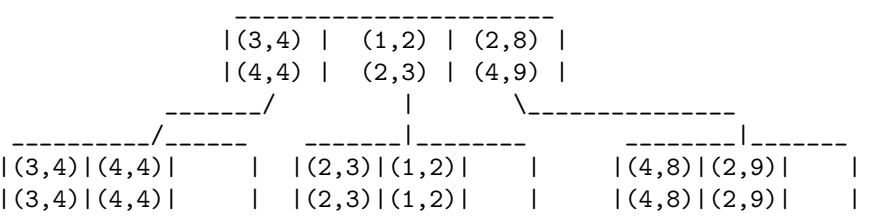
   Form a balanced 2-d binary search tree using the median of the elements to split, with all the keys at the leaves of the tree. To make things unique, when you have to

   nd a median of a list with an even number of entries, choose the smaller of the two entries as the median. Assume also that when a key is equal to the navigation key, it is added to the left subtree (that is, less than or equal).

4. Consider the following 2-dimensional set of points:

$$(4,8),(3,4),(2,9),(1,2),(2,3),(4,4),(6,7),(7,9),(1,10)$$

Below is an R-tree of order $M = 3, m = 2$, where the minimum number of children (and keys) is 2, formed using linear splitting, with the following variation: when splitting an area defined by points, use as seeds the two points that are farthest in distance from each other. The R-tree already contains the first six points: $(4,8),(3,4),(2,9),(1,2),(2,3),(4,4)$.



(a) (8 pts) Insert the next three points in order, $(6,7),(7,9),(1,10)$, showing the resulting R-tree at each step. Use minimum increase in perimeter as the rule for navigation when selecting which leaf to insert.

(b) (2 pts) For the above tree, show the tree that results after deleting the point (4,8).

5. Consider the integer knapsack problem, described as follows: There are n objects $i = 1,\ldots,n$, each of which has integer size c_i and value V_i. You are given a bag of total integer size C. You are allowed to put objecs in the bag, as long as the total size of the objects is less than or equal to C. The goal is to find the set of objects that you will put in the bag that give you the maximum value.

One way to solve this problem is to use dynamic programming in an interesting, clever way. Define the following function: $T(k,c)$ for non-negative integers k, c is the highest value you can put in a bag of size c when you can only consider objects $0,\ldots,k$. You can compute this function recursively as follows:

$$T(1,c) = \begin{cases} 0 & \text{if } c < c_1 \\ V_1 & \text{if} c \geq c_1 \end{cases}$$

We can compute this recursively now as follows: when you consider an extra object, then either it is optimal to put this in the bag, so the rest of the object in the bag must fit in the leftover space, or it is best to leave this object out. This leads to this recursion:

$$T(k,c) = \begin{cases} T(k-1,c) & \text{if } c < c_k \\ \max\{T(k-1,c), V_k + T(k-1,c-c_k)\} & \text{otherwise} \end{cases}$$

Once you compute $T(n,C)$, this is the optimal value when all objects are considered for a bag of size C.

(a) (5 pts) Consider the following list of objects: Object 1 has value 3, size 2. Object 2 has value 2, size 3. Object 3 has value 3, size 4. Use the above recursive algorithm to compute $T(3,2), T(3,3), T(3,4), T(3,5)$ and $T(3,6)$.

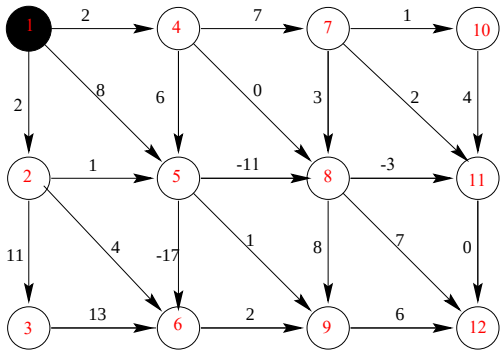
(b) (3 pts) Is this problem (integer knapsack) in class NP? Please explain why.

(c) (3 pts) Provide an expression for the worst case complexity of the above algorithm in terms of n and C. Justify your answer

(d) (2 pts) Is this problem (integer knapsack) in class P? Please explain why.

(e) 2 pts Suppose you are allowed to divide a task and get partial credit for the fraction of the task that you put in the bag. Find the optimal value that you can put into a bag of size 5.

6. Consider a directed acyclic graph with weights on the arcs, with an origin node 1 and a destination node n, as illustrated in the figure on the side. We want to find the ***longest*** path from node 1 to node n.



   (a) (3 pts) Is the decision version of this problem in class NP? If so, explain why.

   (b) (4 pts) Is the decision version of this problem in class P? Explain why or why not.

   (c) (3 pts) Compute the longest path from node 1 to node 8.