CS205: Design and analysis of algorithms

End-semester examination • Maximum marks: 42

Spring 2022

Notes

- Instructor/TA will not be available for clearing doubts during the exam.
- If any assumptions are made, state them and justify.
- There are 9 questions in this question paper. Each carries 6 marks. **Answer only 7 questions. If you answer more than 7 questions, we will evaluate any 7 of them arbitrarily, it may not be the best 7**.
- If it is helpful for your answers, you can use the results we obtained in the lectures, without repeating the proofs discussed in the lectures. But, explicitly state the result that you want to use.
- You are allowed to carry hand-written materials, but no printed materials and electronic gadgets are allowed.

Q1: There is a secret message which contains some small letters in the English alphabet and the white space ' '. The length of the message is 17. You are given with the Huffman encoding of the message along with the frequency of each letter in the message. Your task is to decode the encoded message to obtain the secret message. For this, you have to come up with the Huffman codes for the given characters based on the frequencies. Show the steps by drawing the tree and / or by showing the merges one by one (as done in Assignment 2). To make sure that you decode the encoded string correctly, please follow the following rules:

- Whenever you combine two vertices, one with label string X (having frequency f_x) and another with label string Y (with frequency f_y), then the combined vertex will have label string XY if $f_x < f_y$ or ($f_x = f_y$ and X is lexicographically smaller than Y). In that case, the code of the edge from X to XY will be 0 and that of the edge from Y to XY will be 1. Please note that ' ' (white space) is lexicographically smaller than a, and a is lexicographically smaller than b, and so on. Also note that X is lexicographically smaller than Y if the first letter of X is lexicographically smaller than that of Y.
- As we learned, at every step you choose two vertices of two lowest frequencies and merge them. Assume that there is a tie between two vertices, i.e., with two labels X and Y having the same frequency, then the preference is given to the string which is lexicographically smaller than the other.

The encoded string and the frequencies of characters

00100110111101010100111111010011111010100011000111111000101
' ':3
a:1
b:1
e:3
h:1

l:2
r:1
s:1
t:2
v:1
y:1

The first line of the input data shows the binary string that you have to decode and the rest of the lines specifies the frequencies of the characters in the secret message. **[6 marks]**

Evaluation plan: Correct answer- a=0010,b=0011,e=111,h=0100,l=011,r=0101,s=1000,v=1001, t=101, y=000, space=110. The secret message: all the very best. For merging and finding the string corresponding to each symbol correctly, 5 marks is given. 1 mark is given to find the secret message.

Q2: You are given with a flow network on 8 vertices - see Figure 1. The source is s and the sink is t. Find the maximum flow using Ford-Fulkerson method. Find a minimum (S, T)-cut of the flow network such that the capacity of the cut is equal to the value of the maximum flow that you obtained. **[6 marks]**

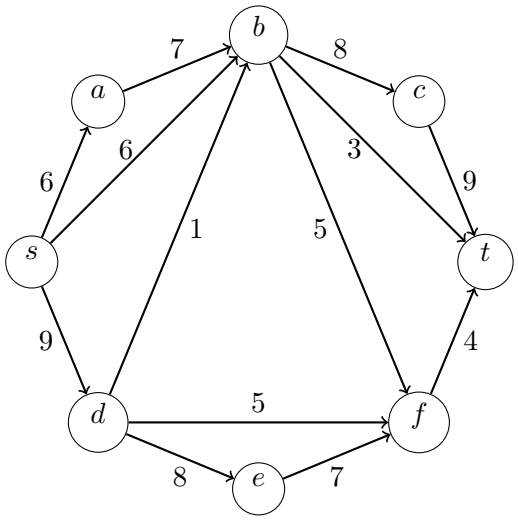

Figure 1:

Evaluation plan:
Maximum flow is 15. Sending a flow of 6 over $sabct$, 2 over $sbct$, 3 over sbt, and 4 over $sdft$ give a flow of 15. A minimum S,T-cut with capacity 15 is $S = \{s, a, b, d, e, f\}$ and $T = \{c, t\}$. You receive 5 marks if you correctly obtain the maximum flow. For every 1 unit that you miss to obtain correctly, you loose 0.5 marks. For example, if you correctly obtain only a flow of 14, then you receive 4.5 marks. For correctly identifying a cut such that the capacity of the cut matches the maximum flow that you obtained, you receive 1 mark.

Q3: Draw all possible red-black trees with exactly 3 internal nodes. The keys are 1,2, and 3. Draw the black nodes as squares/rectangles and the red nodes as circles. **[6 marks]**

Evaluation plan: Two graphs are possible after properly labelling with the keys.

1. root- black, key 2; left child- red, key 1; right child- red, key 3.
2. root- black, key 2; left child- black, key 1; right child- black, key 3.

For each graph 3 marks is given. For each wrong graph 0.5 mark is deducted. If the marks fall below zero, zero mark is given.

Q4: Let T be a red-black tree with at least 4 internal nodes. Prove or disprove that the number of internal black nodes is at least one more than the number of red nodes in T. **[6 marks]**
Evaluation plan: The statement is false. So one should disprove, or give a counter example. Counter example- black root node, it has two black children; each black child has two red children. This tree has four internal red nodes and three internal black nodes. There are infinite number of counter examples. If someone tries to prove, some partial marks will be given based on their understanding.

Q5: As you all have done, Adir has learned a 2-approximation algorithm for Vertex Cover. He claims that, using that algorithm, one can come up with a 2-approximation algorithm for the Independent Set problem. His algorithm is given below.

2-Approx-IS(G)

Step 1: Apply the 2-approximation algorithm for Vertex Cover on G to obtain a vertex cover V' of size at most $2|V^*|$, where V^* is a minimum vertex cover of G.

Step 2: $I' = V(G) \setminus V'$.

Step 3: Return I'.

Answer the following three questions about Adir's algorithm 2-Approx-IS.

(i) Prove or disprove that I' is an Independent Set.

(ii) Prove or disprove that $|I'| \geq |I^*|/2$, where I^* is a maximum independent set of G.

(iii) Using your answers for (i) and (ii), prove or disprove that the algorithm is a 2-approximation algorithm for Independent Set.

[1+3+2 = 6 marks]

Evaluation plan:

(i) Deleting a vertex cover from a graph leaves behind an independent set. Therefore the statement is true. 0.5 marks for trying to prove the statement and 0.5 marks for the correct reasoning.

(ii) One can disprove the statement using a counter-example. Consider a graph G on 6 vertices $\{a, b, c, d, e, f\}$ and three edges $\{ab, cd, ef\}$. Now, the 2-approximation algorithm for vertex cover will return all the vertices of the graph. Therefore I' will be an empty set. The size of a maximum independent set of G is 3. It is not true that $0 >= 3/2$. You receive 0.5 marks for trying to disprove the statement and 2.5 marks for the correct reasoning.

(iii) Since (ii) is wrong, the algorithm is not a 2-approximation algorithm for Independent Set problem. You receive 0.5 marks for trying to disprove and 1.5 for the correct reasoning.

Q6 A perfect matching is a matching in which every vertex is matched, i.e., a set E' of edges of a graph G is a perfect matching of G if no two edges in E' share a vertex and for every vertex v in G, there is an edge in E' incident to v. Design a polynomial-time algorithm which accepts a bipartite graph G as input and finds whether G has a perfect matching or not. You can use any of the algorithms that we discussed in the lectures as a sub-routine. Explain why your algorithm is correct and find the worst-case time complexity of your algorithm. **[6 marks]**
Evaluation plan:

Clearly, a perfect matching has n/2 edges. We have already learned an algorithm to find maximum matching in a bipartite graph. We can solve the mentioned problem by applying the maximum matching algorithm for bipartite graphs and checking whether the obtained matching has $n/2$ edges or not. The complexity of this algorithm is same as that of the maximum matching algorithm for bipartite graph which is $O(VE)$.
Algorithm along with proper explanation gives you 4 marks, explaining why your algorithm works gives you 1 mark and complexity analysis showing the complexity of $O(VE)$ gives you 1 mark. If the complexity analysis is correct but not tight, then you receive 0.5 marks.

Q7 Let $X[1 \dots n]$, $Y[1 \dots n]$ be two arrays, each containing n numbers already in sorted (ascending) order. Give an algorithm, which runs in time $O(\log n)$ in the worst-case, to find the median of all $2n$ numbers in arrays X and Y. Explain the correctness of your algorithm and prove that it runs in $O(\log n)$-time in worst-case. You may assume that all the $2n$ numbers are distinct. You may also assume that the median of $2n$ numbers is the n^{th} smallest number. For example, if $X = [4, 7, 9, 10, 24]$ and $Y = [1, 2, 3, 20, 23]$, then the median of the 10 numbers is 7. **[6 marks]**

Expect this problem in one of your future interviews!
Evaluation plan:
The following is the idea. Let m be the median of $X \cup Y$, that we want to find out. Consider the median x of X and the median y of Y. Assume that $x < y$. Then the first half X_L of elements of X is less than m and the second half Y_R of elements of Y are greater than m. Then m is the median of $X - X_L$ and $Y - Y_R$. The detailed algorithm is given below:
Median-of-2-sorted-arrays(X, Y, i, j, k, ℓ) // The indices start from 1. The algorithm finds the median of the two arrays $X[i \dots j]$ and $Y[k \dots \ell]$. Initially, the algorithm is called with $i = k = 1, j, \ell = |X| = |Y|$.

Step 1: Let $n = j - i + 1$

Step 2: If $n = 1$, return $\min\{X[i], Y[k]\}$.

Step 3: If $n = 2$, return second smallest number in $X[i, i+1] \cup Y[k, k+1]$

Step 4: If n is odd, then:

(i) mid $= (n-1)/2$
(ii) Let $s = \min\{X[i + mid], Y[k + mid]\}$
(iii) Let $t = \max\{X[i + mid], Y[k + mid]\}$
(iv) If $s = X[i + mid]$, then return Median-of-2-sorted-arrays$(X, Y, i + mid, j, k, k + mid)$
(v) If $s = Y[k + mid]$, then return Median-of-2-sorted-arrays$(X, Y, i, i + mid, k + mid, \ell)$

Step 5: If n is even, then:

(i) mid $= n/2 - 1$
(ii) Let $s = \min\{X[i + mid], Y[k + mid]\}$
(iii) Let $t = \max\{X[i + mid + 1], Y[k + mid + 1]\}$
(iv) If $s = X[i + mid]$ and $t = X[i + mid + 1]$, then return $Y[k + mid]$.
(v) If $s = Y[k + mid]$ and $t = Y[k + mid + 1]$, then return $X[i + mid]$.
(vi) If $s = X[i+mid]$, then return Median-of-2-sorted-arrays$(X, Y, i+mid, j, k, k+mid+1)$
(vii) If $s = Y[k+mid]$, then return Median-of-2-sorted-arrays$(X, Y, i, i+mid+1, k+mid, \ell)$

Since, roughly half of the elements are removed in each step, the algorithm runs in $O(\log n)$ time.
A correct algorithm fetches you 3 marks, proper description of the algorithm, explanation for correctness, and the complexity analysis give you one mark each. Since we have learned an algorithm to find the median in linear time, an algorithm which runs in sub-linear time will only give you a non-zero mark.

Q8 An independent vertex cover is a vertex cover which is an independent set. Note that not every graph has an independent vertex cover. For example, the triangle graph has no independent vertex cover. The INDEPENDENT VERTEX COVER (IVC) problem is defined as follows: Given a graph G find whether G has an independent vertex cover or not. Prove that IVC is NP-Complete, or come up with a polynomial-time algorithm to solve it (in this case, explain the correctness of your algorithm and analyse the worst-case time complexity). **[6 marks]**

Evaluation plan: Suppose a graph has an independent vertex cover V'. We know $V(G) \setminus V'$ is an independent set. So the graph is a bipartite graph. To prove this fact 4 marks is given. To come up with a polynomial time algorithm to recognize a bipartite graph, 2 marks is given. Recognizing bipartite graph can be done using Breadth First Search.

Q9 The Alternating Double Coin Game (ADCG) is similar to the Alternating Coin game that we discussed in the class. In ADCG, there are 2 players and there are two arrays C and D of n coins each, where n is an even number. The players take turns in removing one coin each from the two arrays, i.e., in one turn, a player can take a total of two coins - one from C and the other from D. But there is a catch. A player has only two options in a turn - (i) remove the left coin of array X and the right coin of array Y, or (ii) remove the right coin of array X and the left coin of array Y. Here, the left coin of an array is the coin at the least available index of the array, and the right coin is the coin at the largest available index of the array. For example, if $X = [7, 4, 9]$ and $Y = [5, 1, 4]$, then the player has to remove either 7 and 4 or 9 and 5. Therefore, the player gets coins worth 11 or 14 in this turn. Devise an algorithm to maximize the value of the coins collected by the first player. Explain the correctness and analyse the worst-case time complexity of your algorithm. **[6 marks]**

Evaluation plan:

Let $C[1 \dots n]$ and $D[1 \dots n]$ denote the coins. Let $g[i, j, k, \ell]$ denote the maximum that the first player gain if the available arrays are $C[i \dots j]$ and $D[k \dots \ell]$. The dynamic programming formulation of g is given below.

$$g[i,j,k,\ell] = \begin{cases} \max\{C[i] + D[\ell], C[j] + D[k]\}, & \text{if } i = j - 1 \\ \max\{C[i] + D[\ell] + \min\{g[i+2, j, k, \ell-2], g[i+1, j-1, k+1, \ell-1]\}, & \\ C[j] + D[k] + \min\{g[i+1, j-1, k+1, \ell-1], g[i, j-2, k+2, \ell]\}\}, & \text{if} i < j - 1 \end{cases}$$

Note that $i = j$ case does not arise for the first player as n is even. A dynamic programming algorithm for this problem works as follows: Fill a 4-dimensional array $g[i, j, k, \ell]$ using the dynamic programming formulation. The answer will be $g[1, n, 1, n]$. Since the table size is $O(n^4)$ and filling each cell takes constant time, the complexity of the algorithm is $O(n^4)$. Another more efficient approach is to convert this into an Alternating Coin game. Note that when a player chooses i^{th} coin from C she has to choose the $n - i + 1^{\text{th}}$ coin from D. One can come up with a new array of coins Z such that $Z[i] = C[i] + D[n - i + 1]$. Then the same approach for Alternating coin game works. Here, the complexity will be $O(n^2)$.

A correct dynamic programming formulation gives you 3 marks - if the boundary case is missing, then 0.5 marks will be reduced. An explanation of the algorithm, an explanation of why the algorithm works, and the complexity analysis give you one mark each.