NATIONAL UNIVERSITY OF SINGAPORE
SCHOOL OF COMPUTING

Final Assessment for CS3230

December 1, 2021 Time Allowed: 120 minutes

INSTRUCTIONS:

- This paper consists of **FOUR** questions, each worth 25 marks.
- Answer **ALL** questions.
- This is an **OPEN BOOK/NOTES** examination.
- Do **NOT** use the internet or contact others for help while taking the examination. Any evidence otherwise will result in **0** for the entire exam.
- **SUBMISSION**: Please hand-write your solutions. After the time allocated for writing is over, create **exactly one PDF file for each of the FOUR questions** and name your file as ⟨Matric no⟩_Q⟨question no⟩.pdf (e.g., `A324516H_Q3.pdf`). Then, submit each file separately in the corresponding LumiNUS folder (you can see four different folders for each of the FOUR questions inside Files→Final Exam Submissions→Group ⟨group no⟩)

QUESTIONS:

1. **Short Questions.** (25 Marks)

   (a) [**8 Marks**]
   Give an upper bound on $T(n)$ in big O notation, where:
   - $T(n) \leq 1000$ for $n \leq 100$, and
   - $T(n) \leq T(n/2) + O(n \log n)$ for $n > 100$.

   You don't need to give a proof. (As usual, ignore the fact that $n/2$ may be non-integral.)

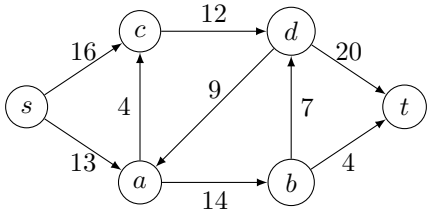
   (b) [**3 Marks for each of (i), (ii), (iii)**]
   Answer true or false. No justification necessary.

   (i) Suppose the amortized cost of an operation op over 7 calls to it is 4. It is possible that for 3 out of the 7 calls, the actual cost of op is 10.

   (ii) You learn about a problem GraphJiraffication that is in NP. You secretly work on the problem for years and finally come up with a polynomial-time algorithm. You have shown P = NP.

   (iii) G is a weighted, undirected graph. Suppose for a node u in the graph, all its incident edges have distinct weights, and e is the smallest weight edge incident on u. Then, any minimum spanning tree of G must contain e.

   (c) [**8 Marks**]
   Consider the flow network shown below. The leftmost node is the source, and the rightmost node is the target. The label on each edge is its capacity.



   Write the sequence of flows, augmenting paths and residual networks during an execution of the Ford-Fulkerson algorithm on this network.

2. **Scheduling Rides**. (25 Marks)

By an unexpected twist of fate, you become the chief operating officer of the Singapore Flyer!

Riders pre-register the day before and specify the time at which they will arrive. Let $t_1 \leq t_2 \leq \cdots \leq t_n$ be the sorted list of times at which the riders arrive. You can choose to set the times $s_1 \leq s_2 \leq s_3 \leq \cdots$ when you start a rotation of the ferris wheel, so that you maximize your earnings from ticket sales.

Rider i who arrives at time t_i starts riding at time s_j, where j is the smallest[1] index such that $s_j \geq t_i$. In this case, rider i pays $1 + \frac{1}{1+s_j-t_i}$ dollars for the ticket. Here, $s_j - t_i$ is the waiting time for rider i before his ride starts. The less the waiting time, the more he pays. If the waiting time is 0, he pays \$2.

The Flyer stops after each full rotation which takes 30 minutes or 0.5 hours. It's not possible to board a moving wheel. So, you must satisfy $s_j - s_{j-1} \geq 0.5$ for all $j > 1$.

As an example, suppose[2] $t_1 = 18, t_2 = 18.25, t_3 = 18.75$. Then, if you start the Flyer at $s_1 = 18, s_2 = 18.75$, then your revenue is $(1 + 1/1) + (1 + 1/1.5) + (1 + 1/1) \approx 5.67$, because the first rider waits for 0 hours, the second for 0.5 and the third for 0. But if you start it at $s_1 = 18.25, s_2 = 18.75$, then you earn $(1 + 1/1.25) + (1 + 1/1) + (1 + 1/1) = 5.80$ because the first rider waits for 0.25 hours, the second for 0 and third for 0. It turns out that the second choice is optimal for this example.

Here is the problem you need to solve: ***Given a set of times*** $t_1 \leq t_2 \leq \cdots \leq t_n$ ***when the riders arrive, what is the maximum possible ticket revenue that could be generated?*** Given your training in CS3230, you turn to dynamic programming.

(a) [**10 Marks**]

For $i = 1, \ldots, n$, let R_i be the maximum revenue that can be generated from the riders arriving at $t_1, \ldots, t_i$ if you start a rotation of the Flyer at time t_i.

Write a recurrence that expresses R_i in terms of R_j for $j < i$. You don't need to prove the correctness of the recurrence, but it should be based on an optimal sub-structure property.

(b) [**5 Marks**]

What is the answer to the original problem (in bold above) in terms of the R_i's?

(c) [**10 Marks**]

Write in pseudocode an algorithm to solve the original problem, using the recurrence formulated in (a). You don't need to prove its correctness.

What is the best running time that you can achieve? Prove your claim.

Note: Do not assume that memoization is automatically available.

[1] We assume there are always seats available, and each rider starts riding as soon as they can.

[2] representing 6 pm, 6:15 pm, 6:45 pm

3. **Breaking Sticks.** (25 Marks)

Your 6-year old nephew is intent on breaking a long stick into n pieces. Let L be the length of the stick. He wants to break the stick at $n-1$ positions that are at distances $d_1, d_2, \ldots, d_{n-1}$ from the left end. Assume $d_1 < d_2 < \cdots < d_{n-1}$. For convenience, define $d_0 = 0$ and $d_n = L$.

Suppose the difficulty of breaking a stick is measured by its length. So, the first break always has difficulty L. Help your nephew find a sequence of breaks that breaks the stick at the desired positions and also minimizes the total difficulty.

For example, suppose $L = 4$, $d_1 = 1$, and $d_2 = 2$. Then breaking at d_2 first and then d_1 has total difficulty $4 + 2 = 6$. But breaking at d_1 first and then d_2 has total difficulty $4 + 3 = 7$. So, the first sequence of breaks is better.

(a) [**5 Marks**]
Consider the greedy algorithm that repeatedly breaks at the right-most break point of the stick. Give a counterexample where this algorithm fails to produce the optimal sequence of breaks.

(b) [**6 Marks**]
Consider the greedy algorithm that finds a break point as close to the center as possible at each step and recurses on the two parts. Give a counterexample where this algorithm fails to produce the optimal sequence of breaks.

(c) [**6 Marks**]
In the rest of the problem, we focus on a dynamic programming formulation. Let $C(i, j)$ denote the minimum total difficulty of breaking the portion of the stick between d_i and d_j at the break points contained in this interval. We want to compute $C(0, n)$.
Consider a particular sub-problem $C(i, j)$. Suppose in an optimal sequence of breaks for this sub-problem, the first break is at position d_k where $i < k < j$. Describe and prove an optimal sub-structure property for $C(i, j)$ in terms of k.

(d) [**8 Marks**]
Use the formulation of part (c) to design an $O(n^3)$ time algorithm for computing $C(0, n)$. Write down a recurrence for $C(i, j)$, and explain clearly how to fill out the dynamic programming table in a bottom-up fashion.
Note: While you may choose to use pseudocode for clarity in your explanation, it is not required. It is not necessary to formally analyze the running time.

4. **Not All are Equal.** (25 Marks)

The NAE-3-SAT problem is the following. Given as input a list of clauses, each on exactly 3 variables, find an assignment such that in each clause, there is at least one true literal and one false literal. For example, if the input instance is:

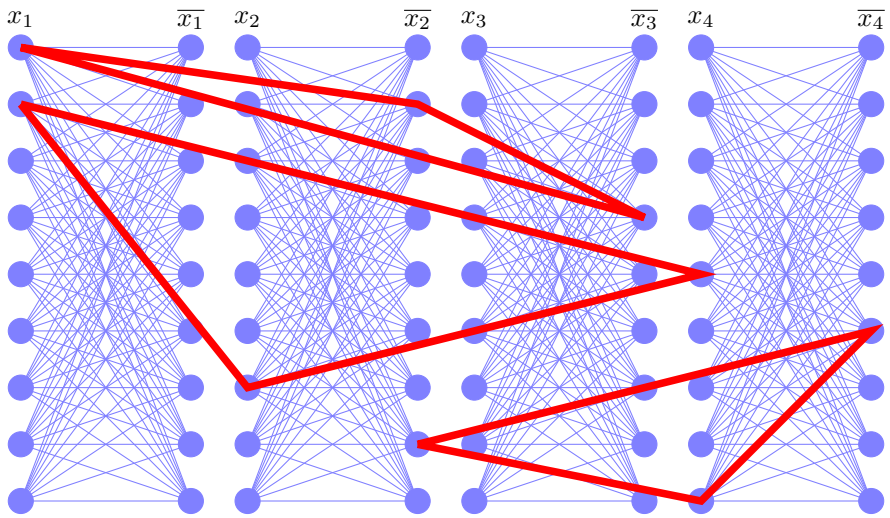
$$(x_1, \overline{x_2}, \overline{x_3}), (x_1, x_2, x_4), (\overline{x_2}, x_4, \overline{x_4})$$

then a satisfying assignment is (x_1, x_2, x_3, x_4) = (TRUE, TRUE, FALSE, FALSE). It is known that NAE-3-SAT is NP-hard.

In this problem, you are asked to show that the MAX-CUT problem is NP-hard by reducing from NAE-3-SAT. The MAX-CUT problem is the following: given as input (G, k) where G is an undirected graph and k is an integer, find whether there is a partition of the vertices of G into two sets S and $\overline{S}$ such that the number of edges with one endpoint in S and another endpoint in $\overline{S}$ is at least k.

Consider the following reduction from NAE-3-SAT to MAX-CUT. Suppose we have an NAE-3-SAT instance $\mathcal{I}_{\text{NAE}}$ with m clauses and n variables. Construct a graph G, where each variable x in $\mathcal{I}_{\text{NAE}}$ is represented with $3m$ vertices labeled x and $3m$ vertices labeled $\overline{x}$, and all $(3m)^2 = 9m^2$ edges are present between vertices labeled x and vertices labeled $\overline{x}$. Additionally, for every clause (a, b, c) in $\mathcal{I}_{\text{NAE}}$ where a, b, c are literals, G contains a triangle with its 3 vertices labeled a, b, and c. These triangles are all disjoint (which is possible since G contains enough copies of vertices with the same label). Finally, let $k = 9m^2n + 2m$ to get an instance of MAX-CUT $\mathcal{I}_{\text{Max}} = (G, k)$.

The example NAE-3-SAT instance above reduces to the following graph:



(a) [**10 Marks**]
Prove that if $\mathcal{I}_{\text{NAE}}$ is a YES-instance of NAE-3-SAT, then $\mathcal{I}_{\text{Max}}$ is a YES-instance of MAX-CUT.

(b) [**15 Marks**]
Prove that if $\mathcal{I}_{\text{NAE}}$ is a NO-instance of NAE-3-SAT, then $\mathcal{I}_{\text{Max}}$ is a NO-instance of MAX-CUT.