Midterm 2

Name:

SID:

Exam Room:

SID of student to your left:

SID of student to your right:

Do not turn this page until your instructor tells you to do so.

- After the exam starts, please *write your name on the front of every page*. We may deduct points if your name is missing from any page. You will not be allowed to fill in your name after time is called.
- For short question, your answers must be written clearly inside the box region. Any answer outside the box will not be graded. For longer question, if you run out of space, you must clearly mention in the space provided for the question if part of your answers are elsewhere.
- Try to answer all questions. Not all parts of a problem are weighted equally. Before you answer any question, read the problem carefully. Be precise and concise in your answers.
- You may use the blank page at the back for scratch work, but we will not look at it for grading.
- You may consult only *two sheet of notes*. Apart from that, you may not look at books, notes, etc. Calculators, phones, computers, and other electronic devices are NOT permitted.
- There are **16** pages (8 double sided sheets) on the exam. Notify a proctor immediately if a page is missing.
- **Any algorithm covered in the lecture can be used as a blackbox.**
- **You have 110 minutes: there are 8 questions on this exam worth a total of 71 points.**

1. **(10 points) True/False**

   Clearly put your answers in the answer box in front of each question. Write T for true, F for false. Each problem will be graded as follows: 1 point for each correct answer and 0 point for blank answer, and -0.25 point for each incorrect answer.

   (a) The maximum weight edge in a graph can not be in any MST.

   (b) $\log^*(65536) = 5$ (All logarithms are base 2).

   (c) The Horn SAT formula $x_1 \implies x_2, \quad x_2 \implies x_1, \quad \bar{x}_1$ is satisfiable.

   (d) An implementation of a dynamic programming problem via memoization does not yield any asymptotic improvements (for any problem instance) in running time with respect to an iterative solution.

   (e) The knapsack problem (without repetition) on n items where for each $i \in \{1, \cdots, n\}$ the weight of the i^{th} item is i can be solved in time polynomial in the input length.

   (f) The region satisfying the equations $0 \le x \le 1, 0 \le y \le 1$, and $y \le x^2$ is convex.

   (g) In a dynamic programming solution, the asymptotic space requirement is always at least as big as the number of unique subproblems.

   (h) In a dynamic programming solution, the asymptotic time complexity is always at least as big as the number of unique subproblems.

   (i) There exists a polynomial time algorithm to find the an approximate minimum set cover (on n items) of size at most $\log_2 n$ times the size of the optimal set cover.

   (j) Huffman encoding can always be used to reduce the size of any document.

2. **(4 points)** Show the tree structure in the union-find algorithm as the following sequence of commands is executed. Use union-by-rank and path compression.

   (a)
```
for $i = \{1, \cdots, 6\}$ {
      MakeSet($i$);
}
union(1,2);
union(3,4);
union(5,6);
union(1,6);
```

   (b)
```
find(6);
```

Name:

3. **(5 points)** Find the Huffman encoding for the following alphabet and set of frequencies. Feel free to draw the tree, but we will only grade the codewords.

$$\{(a,0.12),(b,0.38),(c,0.1),(e,0.25),(f,0.06),(d,0.05),(g,0.01),(h,0.03)\}$$

When you build up your Huffman tree, you should place the branch of lower weight on the left. A left or right branch should respectively correspond to a 0 or 1 in the codeword. Fill the table below with your *final* answers. We will only grade this table. Feel free to use the rest of the page as scratch space.

Symbol	Codewords
a	
b	
c	
d	
e	
f	
g	
h	

4. **(10 points)** Briefly describe an efficient algorithm (and report the running time) for finding a Minimum Spanning Tree in an undirected, connected graph $G = (V, E)$ when the edge weights satisfy:

   (a) **(4 points)** For all $e \in E$, $w_e = 1$.
   Main idea of your algorithm (no pseudocode).

   Time complexity of your algorithm. (Put your justification under the box)

Name:

(b) **(6 points)** For all $e \in E$, $w_e \in \{1,2\}$ (In other words, all edge weights are either 1 or 2).
Main idea of your algorithm (no pseudocode).

Time complexity of your algorithm. (Put your justification under the box)

5. **(6 points) Linear Programming Problem** Given m different types of food, $F_1, ..., F_m$, that supply varying quantities of the n nutrients, $N_1, ..., N_n$, that are essential to good health.

   - Let c_j be the minimum daily requirement of nutrient, N_j.
   - Let b_i be the price per unit of food, F_i.
   - Let a_{ij} be the amount of nutrient N_j contained in one unit of food F_i.

   Write a linear program to plan a meal that supplies all the required nutrients at minimum cost. Make sure to define the variables you use in the linear program.

   (a) **(2 points)** Define the LP variables and what they stand for.

   (b) **(4 points)** Write the objective function and constraints.

6. **(11 points) Coin Game.** Two players play a game. There is a shared array of coins $A[a_1..a_N]$. Players alternate turns to pick either the leftmost or rightmost element of the array. They may not choose an element anywhere in the middle. Once a coin has been picked, it is removed from the array. A player's ”score” is the sum of a_i values of the coins that the player picks. Assuming both players play optimally, what is the maximum score for the starting player?

   (a) **(3 points)** Define your subproblem. (Your subproblem should not be more than 3 lines long.)

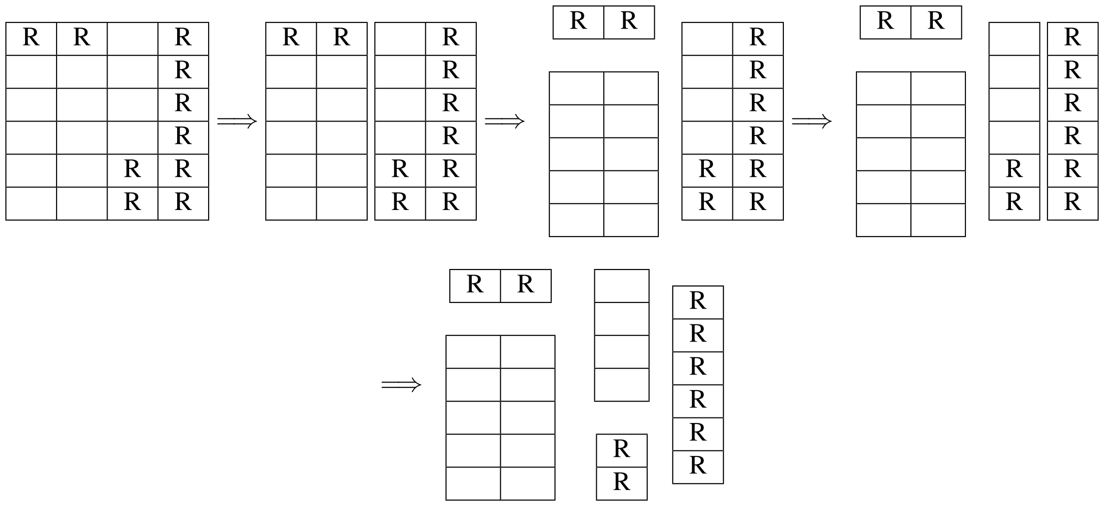
   (b) **(6 points)** Write down the recurrence relation for your subproblems in the box below. (Put your justification under the box.)

   (c) **(1 point)** What are the base cases?

   (d) **(1 points)** Write down the time complexity. (No need for justification)

7. **(10 points) Breaking Chocolate.** There is a chocolate bar consisting of an $m \times n$ rectangular grid of squares. Some of the squares have raisins in them, and you hate raisins. You would like to *break* the chocolate bar into pieces so as to separate all the squares with raisins, from all the squares with no raisins.

   For example, shown below is a 6×4 chocolate bar with raisins in squares marked R. As shown in the picture, one can separate the raisins out in exactly four *breaks*.



   (At any point in time, a *break* is a cut either horizontally or vertically of one of the pieces at the time)

   Design a DP based algorithm to find the smallest number of breaks needed to separate all the raisins out. Formally, the problem is as follows:

   Input: Dimensions of the chocolate bar $m \times n$ and an array $A[i, j]$ such that $A[i, j] = 1$ if and only if the ij^{th} square has a raisin.

   Goal: Find the minimum number of breaks needed to separate the raisins out.

   (a) **(3 points)** Define your subproblem. (Your subproblem should not be more than 3 lines long.)

(b) **(6 points)** Write down the recurrence relation for your subproblems in the box below. (Put your justification under the box; There is a lot more space than you need, try to keep the answers as precise and concise as possible)

(c) **(1 points)** Write down the time complexity in the box below. (No need for justification)

8. **(15 points) Balloon Popping Problem.** You are given a sequence of n-balloons with each one of a different size. If a balloon is popped then it produces noise equal to $s_{left} \cdot s_{popped} \cdot s_{right}$, where s_{popped} is the size of the popped balloon and s_{left} and s_{right} are the sizes of the balloons to its left and to its right. If there is no balloons to the left then we set $s_{left} = 1$ and similarly, if there are no balloons to the right then we set $s_{right} = 1$, while calculating the noise produced.

   After popping a balloon, the balloons to its left and right become neighbors. (Note that the total noise produced depends on the order in which the balloons are popped.)

   Design a dynamic programming algorithm to compute the the maximum noise that can generated by popping the balloons.

   Example:
   Input (Sizes of the balloons in a sequence): ④⑤⑦
   Output (Total noise produced by the optimal order of popping): 175

   Walkthrough of the example:

   - **Current State** ④⑤⑦
     Pop Balloon ⑤
     Noise Produced $= 4 \cdot 5 \cdot 7$

   - **Current State** ④⑦
     Pop Balloon ④
     Noise Produced $= 1 \cdot 4 \cdot 7$

   - **Current State** ⑦
     Pop Balloon ⑦
     Noise Produced $= 1 \cdot 7 \cdot 1$

   - **Total Noise Produced** $= 4 \cdot 5 \cdot 7 + 1 \cdot 4 \cdot 7 + 1 \cdot 7 \cdot 1$.

   (a) **(4 points)** Define your subproblem. (Your subproblem should not be more than 3 lines long.)

(b) **(1 point)** What are the base cases?

(c) **(4 points)** Write down the recurrence relation for your subproblems in the box below. (Put your justification under the box; There is a lot more space than you need, try to keep the answers as precise and concise as possible). **Don't miss part (d) on the next page!**

(d) **(6 points)** Write down your pseudocode.

Name:

DO NOT TEAR ANY PAGES OFF YOUR EXAM.

(Extra page for scratch work.)

DO NOT TEAR ANY PAGES OFF YOUR EXAM.

(Extra page for scratch work.)