Assignment 7

CS 577 *Greedy Methods* Fall 2024

Answer the questions in the boxes provided on the question sheets. If you run out of room for an answer, add a page to the end of the document.

Related Readings: `http://pages.cs.wisc.edu/~hasti/cs240/readings/`

Name: _____ Wisc id: _____

- **Purpose of Homework:**
  - Algorithm design and analysis, like any skill, can only be developed through consistent practice and feedback. Whether it's cooking, playing basketball, integration, gardening, interviewing, or teaching, theoretical knowledge alone is not sufficient. The comfortable feeling of "Oh, sure, I get it" after following a well-presented lecture or hearing a TA explain a homework solution is a seductive, yet dangerous trap. True understanding comes from doing the thing—by actually solving the problems yourself.
  - The homework assignments are your opportunity to practice. Lectures, textbooks, office hours, labs, and guided problem sets are designed to build intuition and provide justification for the skills we want you to develop. However, the most effective way to develop those skills is by attempting to solve the problems on your own. The process is far more important than the final solution.
  - Expect to get stuck. It's normal to have no idea where to start on some problems. That's why you have access to a textbook, lecture slides, and discussions. The journey of wrestling with the problem is an essential part of the learning process.

Scoring Guidelines

You do not need to solve all the exercises!!!

1) 10 points 2a) 2.5 points	2b) 2.5 points	2c) 5 points	
3a) 5 points	3b) 5 points	3c) 5 points 4a) 2.5 points	4b) 2.5 points
5a) 5 points	5b) 5 points 6) 10 points		
7a) 7.5 points	7b) 2.5 points 8a) 2.5 points	8b) 2.5 points	
9a) 2.5 points	9b) 2.5 points	9c) 7.5 points	9d) 7.5 points
10) 10 points 11) 5 points			
12-13a) 2.5 points	12-13b) 12.5 points	12-13c) 7.5 points	12-13d) 2.5 points
14a) 5 points	14b) 5 points 15) 15 points		
16) 5 points 7a) 2.5 points	17b) 2.5 points		
18) 20 points			

The total number of points is **190**, without any rescaling. Therefore, solving all exercises is not required to obtain a high score. To guide your focus, I suggest prioritizing the following exercises: $S = \{3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14\}$. Solving these will give you an additional **30% bonus**.
Your final score can be calculated as:

$$\text{Score} = \left\lceil \sum_{k=1}^{18} \text{Exercise}_k + 0.3 \times \sum_{k \in S} \text{Exercise}_k \right\rceil$$

With the bonus, the total score can potentially reach the maximum limit of 222 points.

Homework Guidelines

- **Collaboration and Academic Integrity:**
  - You are encouraged to work together on homework problems, but you must list everyone you worked with for each problem.
  - You must write everything in your own words and properly cite every external source you use, including ideas from other students. The only sources that you are not required to cite are the official course materials (lectures, notes, homework solutions).
  - Plagiarism is strictly prohibited. Using ideas from other sources or people without citation is considered plagiarism. Copying verbatim from any source, even with citation or permission, is also considered plagiarism. Don't cheat.
- **Submission Instructions:**
  - Submit your homework solutions as PDF files on Gradescope. Submit one PDF file per numbered homework problem.
  - Gradescope will not accept other text file formats such as plain text, HTML, LaTeX source, or Microsoft Word (.doc or .docx).
  - Homework submitted as images (.png or .jpg) will not be graded.
  - Each submitted PDF file should include the following information prominently at the top of the first page: [your full name]_[course title]_[homework assignment number].pdf
- **Solution Writing:**
  - When writing an algorithm, a clear description in English is sufficient. Pseudo-code is not required.
  - Ensure that your algorithm is correct by providing a justification, and analyze the asymptotic running time of your solution. Even if your algorithm does not meet the requested time bounds, you may receive partial credit for a correct, albeit inefficient, solution.
  - Pay close attention to the instructions for each problem. Partial credit may be awarded for incomplete or partially correct answers.

In one or two sentences, describe what a greedy algorithm is. Your definition should be informal, something you could share with a non computer scientist.

Dividing the Minoan Labyrinth - Wedge Requests

1. After the fall of the Minotaur, King Minos decided to divide the legendary labyrinth into wedge-like sections as a symbol of unity and triumph. These wedges were to be distributed to citizens as tokens of heritage. Each citizen, including noble warriors and farmers, submitted requests for a specific angular wedge of the labyrinth, defined by two angles. However, some of the wedge requests overlapped, making it impossible to satisfy everyone. King Minos is now tasked with finding the maximum number of requests that can be satisfied without overlapping sections. Given this problem, describe and analyze a quadratic algorithm to find the maximum number of citizens whose wedge requests can be satisfied.
Hint: The solution should not depend on the rotation of the labyrinth, and angles are not assumed to be integers. There is a research paper suggesting a solution in $O(n \log n)$ time. Anyone who creates slides and presents the solution will receive 40 bonus points.

Scheduling problems

There are many different problems all described as "scheduling" problems. In the following questions, pay attention to the details of the problem setup, as they will change each time!

2. Let's start with the standard setting of interval scheduling problem with one processor and multiple jobs.
   (a) Let each job have a start time, an end time, and a value. We want to schedule as much value of non-conflicting jobs as possible. Use a counterexample to show that Earliest Finish First (the greedy algorithm we used for jobs with all equal value) does NOT work in this case.
   (b) *Kleinberg, Jon. Algorithm Design (p. 191, q. 7)* Now let each job consist of two durations. A job i must be preprocessed for p_i time on a supercomputer, and then finished for f_i time on a standard PC. There are enough PCs available to run all jobs at the same time, but there is only one supercomputer (which can only run a single job at a time). The completion time of a schedule is defined as the earliest time when all jobs are done running on both the supercomputer and the PCs. Give a polynomial time algorithm that finds a schedule with the earliest completion time possible.
   (c) Prove the correctness of your algorithm from part (b).

Selecting Volunteers for MadHacks Mentorship Sessions

3. Consider n volunteers who will work at MadHacks, each covering a specific shift over the hackathon weekend. The shift for each volunteer i is defined by their start time s_i and end time f_i, corresponding to the continuous time interval $[s_i, f_i)$. The MadHacks organizing team needs to select the minimum-size group of volunteer representatives for obligatory mentorship sessions, ensuring that for each volunteer i, there is at least one representative whose shift overlaps with that of i.
   (a) *Show that the following greedy choice is not optimal:*
   As long as there are uncovered intervals, choose the largest interval that can cover them.
   (b) *Design an efficient algorithm to select the optimal group of representatives*
   (c) *Explain the correctness of your approach along with its computational complexity.*

Scheduling Classes at Hogwarts

4. Harry and his friends need to cast protective spells over specific locations on the 1-D Marauder's Map, marked by a set $X = \{x_1, \ldots, x_n\}$. However, each spell only covers a unit-length range. Their goal is to use the fewest number of spells to ensure every marked point on the map is protected. Help them figure out a simple strategy to achieve full coverage with the minimum number of spells.
   (a) Describe your algorithm and analyze its running time.
   (b) In this exercise, we will show the correctness of the greedy choice property and the optimal substructure. Prove the following claims:
      1. **Claim: There is an optimal solution that includes the first greedy choice.**
      2. **Claim: Hogwarts scheduling has the optimal substructure property** [1]

Event Recording

During the progression of a scientific experiment, n significant events are expected to occur, and we want to record their impact. We know that these events will occur at times $x_1, x_2, \ldots, x_n$, where $x_1 < x_2 < \cdots < x_n$. To record the impact of these events, we have access to $m \geq n$ volunteers. Each volunteer i can observe the experiment during the time interval $[s_i, f_i)$, and can record the impact of an event j, under the assumption that $x_j \in [s_i, f_i)$. Given that each volunteer can record the impact of at most one event, we aim to compute a subset of volunteers who can record the impact of all the events (or if the volunteers are insufficient, to verify this).

5. Design an efficient algorithm for this problem and justify the correctness and computational complexity of your algorithm.

Skiers & Skis

6. In Wisconsin, many people enjoy skiing at Cascade Mountain. Suppose there are n skiers with heights given in an array $P[1..n]$, and n pairs of skis with different sizes given in an array $S[1..n]$. Your task is to design an efficient algorithm that assigns a pair of skis to each skier, such that the difference between the skier's height and the ski size is minimized on average. The goal is to find a permutation σ such that the expression:

$$\min_{\sigma \in \text{Permutations}[n]} \frac{1}{n} \sum_{i=1}^{n} |P[i] - S[\sigma(i)]|$$

Motorcyclist Turbo Speeding

7. A motorcyclist is traveling from city A to city B through intermediate cities $1, 2, \ldots, n$. The (only) road that connects each city i with the next city $i + 1$ has length ℓ_i km and a speed limit u_i km/h. The motorcyclist is usually law-abiding and follows the speed limits. However, today he is running late and risks missing the start of a very important event in city B. He travels each segment at the maximum allowed speed u_i unless he decides to exceed the speed limit. Additionally, he has decided, for the first and last time, to exceed the speed limit by v km/h on some selected segments $i_1, i_2, \ldots$, but the total time spent violating the speed limit must not exceed a total time T. He needs to select on which segments $i_1, i_2, \ldots$ of the route to speed and by how much $u_{i_1} + v, u_{i_2} + v, \ldots$ such that the total time spent speeding is T and the arrival time in city B is minimized.

[1] In computer science, a problem is said to have optimal substructure if an optimal solution can be constructed from optimal solutions of its subproblems.

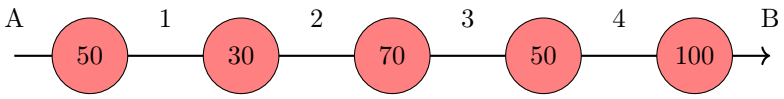

Figure 1: Example of a route and speed limits for the problem.

For example, in Figure 1, we see a route from city A to city B with 4 intermediate stops. The lengths of the corresponding segments are $\ell_0 = 7\,\text{km}, \ell_1 = 12\,\text{km}, \ell_2 = 14\,\text{km}, \ell_3 = 20.5\,\text{km}, \ell_4 = 10\,\text{km}$, and the speed limits are $u_0 = 50\,\text{km/h}, u_1 = 30\,\text{km/h}, u_2 = 70\,\text{km/h}, u_3 = 50\,\text{km/h}, u_4 = 100\,\text{km/h}$. Thus, without exceeding the speed limit, the motorcyclist needs 1 hour and 15 minutes (1.25 hours) to reach city B. If now he exceeds the speed limit by 20 km/h for 24 minutes (0.4 hours), then he can travel at 70 km/h on the segment $(A, 1)$, at 50 km/h on the segment $(1, 2)$, and at 70 km/h on the first 4.2 km of the segment $(3, 4)$. The total duration of speeding is 24 minutes, and the motorcyclist now only needs about 1 hour (1.026 hours exactly) to reach city B.

(a) Design an efficient algorithm to optimally distribute the speeding time among the segments where the motorcyclist exceeds the speed limit. Justify the correctness and computational complexity of your algorithm.

(b) What changes in the problem (and in the algorithm) if the motorcyclist decides to exceed the speed limit by a factor $a > 1$ instead of a fixed amount v km/h?

Efficient Studying

You are taking a very large number of classes, indexed 0 through $n - 1$. For each class i, you have estimated a maximum amount of time g_i that you can study for this class. You have also estimated for each class the benefit b_i that would result from studying for this maximum amount of time. Because these are just estimates, the values g_i and b_i will be integers less than 500. Now, you have T time left before spring break, and you will spend it studying nonstop. Of course, you will not be able to study the goal amount of time g_i for each of your classes. However, for each class you can study for any amount of time $t \leq g_i$, producing a benefit of $t \cdot \frac{b_i}{g_i}$.

8. Your goal is to find an efficient algorithm to determine the maximum (over all possible allocations of your time to your classes) total (additive) benefit you can obtain from studying. You may only study for one class at once. You may not study a negative amount of time for any class.

   (a) Describe your greedy algorithm to solve the problem.

   (b) Prove the correctness and the justify the efficiency of your algorithm.

Cell Phone Towers

9. *Kleinberg, Jon. Algorithm Design (p. 190, q. 5)*

   (a) Consider a long straight road with houses scattered along it. We want to place cell phone towers along the road so that every house is within four miles of at least one tower. Give an efficient algorithm that achieves this goal using the minimum possible number of towers.

   (b) In another rural village, the houses that need coverage are located around a circular lake with a radius much larger than that of the company's towers. Obviously, the company cannot place towers inside the lake. Formulate an efficient algorithm to find the optimal positions for the towers. Justify the correctness and the computational complexity of your algorithm.
   Note: Since the radius of the lake is much larger than that of the towers, you can assume that the portion of the circle covered by one tower will be a circular arc of length $2k$ meters, centered at the tower.

(c) Prove the correctness of your algorithm for linear village.

(d) Prove the correctness of your algorithm for lake village.

Transportation Problems

10. A driver needs to travel from city x to city y with their car, which consumes fuel and has a limitation on the fuel it can store, k kilometers. The driver has a map that shows all the gas stations along the route and the distances between them. The goal is to calculate a refueling plan that minimizes the total number of refueling stops (while ensuring that the distance between two consecutive refuelings does not exceed k kilometers, so the car doesn't run out of fuel). Develop the most efficient algorithm possible to compute an optimal refueling plan. Determine the computational complexity and prove the correctness of your algorithm. Note: You may assume that two consecutive gas stations are at most k kilometers apart.

11. *Kleinberg, Jon. Algorithm Design (p. 189, q. 3).*

    You are consulting for a trucking company that does a large amount of business shipping packages between New York and Boston. The volume is high enough that they have to send a number of trucks each day between the two locations. Trucks have a fixed limit W on the maximum amount of weight they are allowed to carry. Boxes arrive at the New York station one by one, and each package i has a weight w_i. The trucking station is quite small, so at most one truck can be at the station at any time. Company policy requires that boxes are shipped in the order they arrive; otherwise, a customer might get upset upon seeing a box that arrived after his make it to Boston faster. At the moment, the company is using a simple greedy algorithm for packing: they pack boxes in the order they arrive, and whenever the next box does not fit, they send the truck on its way.

    Prove that, for a given set of boxes with specified weights, the greedy algorithm currently in use actually minimizes the number of trucks that are needed. Hint: Use the stay ahead method.

Card & Universe Battles

12. *In a children's card game, player A takes the "positive" cards, and player B takes the "neutral" cards. Each "positive" card i has a weight $a_i \geq 0$ and a value $v_i \geq 0$, and each "neutral" card j has a weight $b_j \geq 0$. Player A knows their own cards and the cards of player B and must compute a one-to-one correspondence between the cards to maximize the value of the "positive" cards that A wins. If a "positive" card i is matched with a "neutral" card j, A wins the "positive" card if $a_i > b_j$, otherwise they lose the card.*
    Design an efficient algorithm that computes an optimal matching of cards for player A. Justify the correctness and computational complexity of your algorithm.

13. *In a galaxy far, far away, the Empire, with the goal of destroying the planets controlled by the rebels, has constructed small Death Satellites. Due to poor design, however, these satellites can target only one planet at a time. Each Satellite has a power f_i. The Rebels have installed a shield on each of their planets that can withstand an attack up to power s_j, and each planet j of the Rebellion has a value v_j to the Empire. If planet j (with shield strength s_j) is attacked by a Satellite with power f_i, and $f_i > s_j$, the Empire will win and gain value v_j; otherwise, the Empire gains nothing. The goal of the Empire is to match its n Satellites with the planets in such a way as to maximize the total value obtained.*
    Design an efficient algorithm to compute an optimal matching of Satellites to planets for the Empire. Justify the correctness and the computational complexity of your algorithm.

    (a) Please describe mathematically why Card and Star Wars Battle are not the same.

    (b) Describe and prove the optimality of your greedy algorithm for Card Battle

    (c) Describe and prove the optimality of your greedy algorithm for Star Wars Battle

    (d) **Finally, how different are the two problems?**

Greedy is not optimal but not bad

In this section, we study the performance of greedy method if it is not optimal.

14. We start by exploring a famous NP-complete problem in the worst case the maximum weight independent set problem in two settings: (1) on a tree, and (2) on a sequence of integers.

    (a) *Part 1: Maximum Weight Independent Set in a Tree*:
    Given a set of vertices $I \subseteq V$ of an undirected graph $G(V, E)$, the set is called independent if no two vertices in I are connected by an edge. In this part, we consider a tree $T(V, E)$ where each vertex $v \in V$ has a weight $w(v) \geq 0$. The objective is to compute an independent set in T with the maximum total weight.

    - Consider a simple algorithm that partitions the vertices of T into two independent sets I_0 and I_1 (explain how this can be done in linear time) and returns the set (between I_0 and I_1) with the larger total weight. Find an example where this greedy algorithm fails to compute the optimal solution. However, the solution computed by the greedy algorithm guarantees a significant fraction of the optimal total weight of an independent set in T. What is this fraction, and why?

    (b) *Part 2: Maximum Weight Independent Set in a Sequence*:
    Given a sequence $X = (x_1, \ldots, x_n)$ of n positive integers, a set of indices $I \subseteq \{1, \ldots, n\}$ is called independent if for every pair of indices $i, j \in I$, $|i - j| > 1$, meaning that I contains no consecutive indices. The weight $W(J)$ of a set of indices $J \subseteq \{1, \ldots, n\}$ is defined as the sum of the corresponding values x_i, i.e.,

    $$W(J) = \sum_{i \in J} x_i.$$

    The goal is to compute an independent set of maximum weight for the given sequence X.

    - Suppose we use the following greedy algorithm: Initially set $I = \emptyset$ and $J = \{1, \ldots, n\}$. As long as $J \neq \emptyset$, add the index $j \in J$ with the largest value x_j to I, and remove from J the indices $j-1$, j, and $j+1$. The result is the set I. Provide an example where the above greedy algorithm fails to compute the optimal solution. Prove that the independent set I computed by the above greedy algorithm has a weight of at least half of the weight of the optimal independent set.

15. A chessboard with 4 rows and n columns is given, where each square has a monetary amount. There are $2n$ stones, and each stone can be placed on a square of the chessboard to collect the corresponding value. We aim to place some or all of the stones on the chessboard to maximize the total amount collected, but stones cannot be placed on squares that are adjacent either horizontally or vertically (diagonal adjacency is allowed).

    - Consider the greedy algorithm that, as long as possible, collects the highest available value on a square that is not adjacent to any square where a stone has already been placed. Find an example where the greedy algorithm fails to compute the optimal solution.
    - Nonetheless, the greedy algorithm guarantees the collection of a significant percentage of the total amount collected by the optimal solution. What is this percentage, and why?

The Library Problem

A student is given a mandatory study schedule for the next T days. According to the schedule, the student reads one subject each day. The total number of subjects is n, with $n < T$. The student may study some subjects for more than one day, and not necessarily on consecutive days (e.g., Day 1: Algorithms, Day 2: Languages, Day 3: Databases, Day 4: Algorithms, Day 5: Languages, Day 6: Algorithms, Day 7: Algorithms, Day 8: Databases, Day 9: Databases, etc.). Each subject requires borrowing a specific book from the library. The student is not allowed to have more than k, $k < n$, books checked out at the same time. Additionally, the student is not allowed to borrow more than one book per day. Therefore, on any

day that the student needs a book and does not already have it, they must go to the library to borrow it (only that book; no other books can be borrowed on the same day). If the student already has k books checked out, they must choose which book to return. Each visit to the library takes time away from the student's study time. The goal is to determine a book return policy that minimizes the number of library visits.

16. Formulate the most efficient algorithm/policy for returning books that minimizes the number of library visits and prove that your algorithm indeed computes the optimal solution. What is the runtime of your algorithm? Can you relate this library problem to some (important) practical application?

Optimal Cache

17. In class, we saw that an optimal greedy strategy for the paging problem was to reject the page the furthest in the future (FF). The paging problem is a classic online problem, meaning that algorithms do not have access to future requests. Consider the following online eviction strategies for the paging problem, and provide counter-examples that show that they are not optimal offline strategies.[2]
    (a) FWF is a strategy that, on a page fault, if the cache is full, it evicts all the pages.
    (b) LRU is a strategy that, if the cache is full, evicts the least recently used page when there is a page fault.

Lower Bounds in Greedy Algorithms

Recall that almost every greedy algorithm we have encountered operates by performing a linear scan after sorting the input based on a specific greedy criterion. In this final exercise, we will demonstrate that there are cases where we can prove that no better algorithm exists for certain problems, particularly in the case of comparison-based algorithms.

18. Every algorithm that constructs a Huffman tree using only comparisons requires at least $\Omega(n \log n)$ comparisons in the worst case. *Hint: Reduce the Sorting Problem to Huffman Tree Construction.*

[2] An interesting note is that both of these strategies are k-competitive, meaning that they are equivalent under the standard theoretical measure of online algorithms. However, FWF really makes no sense in practice, whereas LRU is used in practice.