Algorithm Design - First Midterm
16-01-2025, *Time: 120 minutes*
Prof. Aris Anagnostopoulos, Prof. Stefano Leonardi, Dr. Simone Fioravanti

Instructions:

- You are given the question sheet and 4 sheets for your answers. The problems are 4.
- Before you start, write on the top of each sheet (1) your full name, (2) your student ID (*matricola*) and (3) the sheet number (1, 2, 3, and 4).
- If you use additional sheets as rough draft (*brutta*) for calculations, etc., you must hand it, along with the rest of the exam. You can keep the question sheet.
- Hand in the sheets in order **without attaching the sheets with each other.**
- When your pages are full, you can ask for extra pages; make sure that you also write there your name, student id, and the sheet number (5, 6, etc.). You must hand all the sheets that you receive.
- The rooms are small so it may be tempting to copy. Note that if we catch you copying in any way (copying from another student, using your phone, etc.) you will be automatically disqualified.

Problem 1

A trucking company does a large amount of business, shipping packages between stations A and B. 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 station A one by one, and each package i has a weight w_i. There are two requirements: at most one truck can be at the station at any time and the boxes must be shipped in the order they arrive. The company wishes to minimize the number of trucks that are used each day.

(a) (2.5 points) Give the pseudocode and study the computational complexity of the following simple greedy algorithm: pack boxes in the order they arrive, and whenever the next box does not fit in the current truck, send the truck on its way and call another one.

(b) (5 points) Prove that the greedy algorithm is optimal.

Problem 2

Let A be an array of integers of length n i.e. $A[i] \in \mathbb{Z}$ for $i = 1, \ldots, n$. The problem is to find the contiguous subarray (i.e. a subarray formed by contigous indices) that maximizes the sum of its elements in at most $O(n)$ time.

(a) (3 points) Let $OPT[i]$ be the optimal subarray ending at position i. Prove a suitable recursive formula for these subproblems.

(b) (3 points) Provide an efficient implementation of the dynamic programming algorithm that computes the value of $OPT[i]$. Discuss its space/time complexities.

(c) (1.5 points) How do you compute the actual subarray given the algorithm at point (b)?

Problem 3

Recall the IndependentSet decision problem: Given an undirected graph G and a constant k, does there exist an independent set (a set of nodes that are not connected by edges) of size $\geq k$?

(a) (2.5 points)Give the definition of polynomial reduction from a decision problem A to a decision problem B.

(b) (5 points) Prove that IndependentSet is NP-complete via a reduction from 3-SAT.

Problem 4

Consider the following optimization problem. Max3SAT: Given a CNF formula consisting of clauses $C = C_1, C_2, \ldots, C_k$ and literals $X = x_1, x_2, \ldots, x_n$ such that each clause involves exactly 3 literals, find a truth assignment that satisfies as many clauses as possible.

(a) (2.5 points) Provide a randomized approximation algorithm for this problem and study its time complexity.

(b) (5 points) Show that the expected approximation ratio is at least $7/8$.

Problem 1

A trucking company does a large amount of business, shipping packages between stations A and B. 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 station A one by one, and each package i has a weight w_i. There are two requirements: at most one truck can be at the station at any time and the boxes must be shipped in the order they arrive. The company wishes to minimize the number of trucks that are used each day.

(a) (2.5 points) Give the pseudocode and study the computational complexity of the following simple greedy algorithm: pack boxes in the order they arrive, and whenever the next box does not fit in the current truck, send the truck on its way and call another one.

(b) (5 points) Prove that the greedy algorithm is optimal.

a) EACH PACK MUST BE $w_i \leq W$

```
GREEDY-ALG:
  TRUCKS = 1
  LOAD = 0

  FOR i = 1 .. n:
    IF LOAD + W[i] ≤ W
       LOAD += W[i]
    ELSE
       TRUCKS += 1
       LOAD = W[i]

  RETURN TRUCKS
```

n = PACKS

O(n): WE SEE ALL PACKS ONE TIME

b) FORALL OPTIMAL SOLUTION S, AFTER K TRUCKS, S CAN'T HAVE SENT MORE PACKS THAN GREEDY.

INDUCTION:

K=1. GREEDY LOAD ON THE FIRST TRUCK THE MAXIMUM WEIGHT IN PACKS P WITHOUT EXCEEDING W. AN OPT SOLUTION S CAN'T HAVE LOADED P+1 PACKS, BECAUSE THIS WOULD EXCEED W.

K-1. LET g_K BE THE INDEX OF THE LAST PACK ON THE K^{TH} TRUCK. IN EVERY SOLUTION, THE K^{TH} TRUCK HAVE TO START FROM THE NEXT PACK TO THE ONE ALREADY SHIPPED. SO S AT MOST CAN START THE K^{TH} TRUCK FROM AN INDEX $\leq g_{K-1}+1$ (IT CAN'T SKIP SOME PACKS) BUT THE GREEDY, STARTING FROM $g_{K-1}+1$, LOAD AGAIN THE MAXIMUM THAT THE TRUCK CAN HANDLE ($\leq W$). SO $S_K \leq g_K$

THE GREEDY STAYS AHEAD.

Problem 2

Let A be an array of integers of length n i.e. $A[i] \in \mathbb{Z}$ for $i = 1, \ldots, n$. The problem is to find the contiguous subarray (i.e. a subarray formed by contigous indices) that maximizes the sum of its elements in at most $O(n)$ time.

(a) (3 points) Let $OPT[i]$ be the optimal subarray ending at position i. Prove a suitable recursive formula for these subproblems.

(b) (3 points) Provide an efficient implementation of the dynamic programming algorithm that computes the value of $OPT[i]$. Discuss its space/time complexities.

(c) (1.5 points) How do you compute the actual subarray given the algorithm at point (b)?

a) A SUBARREY THAT ENDS IN i CAN:

- EXTEND THE OPT THAT ENDED IN $i-1$, ADDING $A[i]$

OR

- RESTART FROM ZERO IN i, TAKING ONLY $[i.i]$

$OPT[i] = max(A[i], OPT[i-1] + A[i])$

b)

```
CURRENT = A[1]
BEST = A[1]

FOR i = 2..n
   CURRENT = max (A[i], CURRENT + A[i])
   BEST = max (BEST, CURRENT)
RETURN BEST
```

O(n)

c)

```
CURRENT = A[1]
BEST = A[1]

START = 1
BESTSTART = 1
BESTEND = 1

FOR i = 2..n:
  IF A[i] > CURRENT + A[i]
     CURRENT = A[i]
     START = i
  ELSE
     CURRENT += A[i]

  IF CURRENT > BEST
     BEST = CURRENT
     BESTSTART = START
     BESTEND = i

RETURN (BEST, BESTSTART, BESTEND)
```

Problem 3

Recall the IndependentSet decision problem: Given an undirected graph G and a constant k, does there exist an independent set (a set of nodes that are not connected by edges) of size $\geq k$?

(a) (2.5 points)Give the definition of polynomial reduction from a decision problem A to a decision problem B.

(b) (5 points) Prove that IndependentSet is NP-complete via a reduction from 3-SAT.

a) $A \leq_p B$ IF $\exists$ A FUNCTION f CALCULABLE IN POLYNOMIAL TIME SUCH THAT FOR EACH INSTANCE $a \rightarrow a \in A \Leftrightarrow f(a) \in B$.
WE TRANSFORM AN A INSTANCE IN A B INSTANCE. SO IF WE CAN RESOLVE B, WE CAN ALSO RESOLVE A.

b) FIRST OF ALL, INDIP SET IS IN NP SINCE WE CAN CHECK, IN POL TIME, THAT A SET S, OF SIZE AT LEAST K, HAS NODES NOT CONNECTED BY EDGES.

NOW 3 SAT $\leq_p$ INDIP SET:

WE CONSTRUCT A GRAPH G, AND FOR EACH CLAUSE C_i WE CREATE 3 CONNECTED NODES. WE ADD AN EDGE BETWEEN TWO CONFLICTING NODES (x AND $\neg x$), BECAUSE WE CAN CHOOSE ONLY ONE. FOR EACH CLAUSE C_i ONLY ONE NODE IS CHOOSED.

THERE EXISTS A TRUTH ASSIGNMENT THAT SATISFY K CLAUSES IFF THERE EXISTS AN INDIPENDENT SET OF SIZE K.

INDIPENDENT SET IS NP-COMPLETE

Problem 4

Consider the following optimization problem. Max3SAT: Given a CNF formula consisting of clauses $C = C_1, C_2, \ldots, C_k$ and literals $X = x_1, x_2, \ldots, x_n$ such that each clause involves exactly 3 literals, find a truth assignment that satisfies as many clauses as possible.

(a) (2.5 points) Provide a randomized approximation algorithm for this problem and study its time complexity.

(b) (5 points) Show that the expected approximation ratio is at least 7/8.

a) WE SET EACH VAR x_i TO 0 OR 1 WITH PROBABILITY $\frac{1}{2}$. WE COUNT HOW MANY CLAUSES HAVE BEEN SATISFIED AND WE RETURN THE FOUND ASSIGNMENT

$O(n)$ FOR ALL ASSIGNMENT
$O(k)$ FOR EACH CLAUSES
} $O(n+k)$

b) LET DENOTE Z BE THE SUM OF THE RANDOM VARIABLES

$$Z_i \begin{cases} 1 & \text{IF } C_i = 1 \\ 0 & \text{IF } C_i = 0 \end{cases} \qquad Z = Z_1 + Z_2 + \cdots + Z_k$$

$E[Z_i]$ IS THE EXPECTED VALUE THAT C_i IS SATISFIED.

THE PROBABILITY THAT C_i IS NOT SAT IS $(\frac{1}{2})^3 = \frac{1}{8}$

SO $E[Z] = E[Z_1] + \cdots + E[Z_k] = 1 - \frac{1}{8} = \frac{7}{8}k$