Homework 3

Graded

Student

Kulvir Singh

Total Points

30 / 30 pts

Question 1

Q1 **10** / 10 pts

✔ **– 0 pts** Correct

– 1 pt [Algorithm] At most 2 issues.

– 3 pts [Algorithm] 3 or more issues.

– 3 pts [Algorithm] Incorrect time complexity.

– 1 pt [Proof] Incorrect base case

– 1 pt [Proof] Incorrect inductive assumption

– 1 pt [Proof] Incorrect inductive step

– 1 pt [Proof] Incorrect proof of contradiction.

– 1 pt [Proof] Other minor issues.

Question 2

Q2 **10** / 10 pts

✔ **+ 10 pts** Correct

+ 5 pts Correct algorithm, not correct or incomplete proof

+ 0 pts Incorrect

Question 3

Q3 **10** / 10 pts

✔ **– 0 pts** Correct

– 3 pts Incorrect runtime

– 7 pts Incorrect Algorithm

Question assigned to the following page: 1

ADA HW 3

i) Algorithm

Assume d_0 to be USC and since it's the start point, the value of d_0 become 0.

since we have p miles initially in the car, we can say that

```
stops = 0
remaining_gas = p

for ( i = 0, run i from 0 ⟶ n-1)
{
  # calculate the next stop's distance as follows ⟶
      next_stop = d_{i+1} - d_i

  if (remaining_gas is >= nextstop) then
      remaining_gas = remaining_gas - nextstop
  }
  else
  {
      stops++
      remaining_gas = p;
  }
}
return stops;
```

Question assigned to the following page: 1

Time Complexity Analysis

Assigning initial values, returning, arithmetic operations all work at constant time → O(1)

Loop through all the stops → O(N)

Total TC = O(N)

Proof of Correctness

i) A is a compatible set.

The algorithm always returns the number of stops and even in the worst case scenario, it cannot return more than N stops as given in the question. This makes A a compatible set.

ii) A is an optimal set.

Let there be an optimal solution O which has a better result than A. Then by Mathematical Induction.

Base Case

when there is only (1 stop)

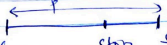
Assume

Optimal sol → 0 stops taken

A sol → 1 stop taken

Question assigned to the following page: 1

However, (by contradiction)
if the optimal sol takes 0 stops, then
the distance between start and
end has to be at most 'p' miles



for A to take the stop, the distance
between $|USC - SM| > p$

∴ A will not take the stop.

⇒ A is compatible with O

~~Mathematical Induction~~ Inductive Step
when there are k stops. ~~(worst case)~~
Assume O → k stops
A → k+1 stops

By contradiction
if the optimal solution takes k stops
then k to Santa Monica can at most
be at distance of 'p'.
& so if from k^{th} stop the distance
to end is at most 'p', according
to Algorithm 'A', there cannot be any
other stops req. to be made.
∴ k+1 stop was never visited.

⇒ A is compatible with O.

Question assigned to the following page: 1

Concluding the induction, we can say that if there exists 2 selections for N step with $0 \rightarrow N$ optimal and $A \rightarrow N+1$, the $N+1$ th step will never be visited.

Hence A is an optimal solution.

Question assigned to the following page: 2

2) given : maximise $\prod a_i^{b_i}$

A	B
a_0	b_0
a_1	b_1
.	.
.	.
.	.

Intuition : To maximise the value of $\prod a_i^{b_i}$ we need to ~~max~~ get the maximum possible exponential values which means that we ~~to~~ max the base and the exponent. This can be done by using ($a_{max}^{b_{max}}$. ——so on)

Algo :

Step 1 → [Sort (A) / Sort (B)] in descending order

Step 2 → ~~store~~ product = 1 (initialise variable to store result)

{ for (i : 0 ⟶ n) // essentially looping through the sorted ~~Arr~~ lists A & B.

~~Store~~ product = product * (A[i] ^ B[i])

}

Step 3 → return product.

Question assigned to the following page: 2

Time complexity Analysis

Step 1 → Sorting A & B.

1 sort operation = $O(n \log n)$

2 sort operations = $O(n \log n) + O(n \log n)$

$= O(2n \log n)$

TC = $O(n \log n)$

Step 2 → Loop through the arrays once but simultaneously.

T.C = $O(N)$

Step 3 → return product TC = $O(1)$

Overall TC = $O(N \log N) + O(N) + O(1)$

TC = $O(n \log n)$

Also, power operation (^) takes a time complexity of $O(\log n)$ which can be dropped as we already have $(n \log n)$

∴

The (^) operation run for n times

total TC can be

$O(N \log N) + O(N \log N) + O(1)$

TC = $O(n \log n)$

Question assigned to the following page: 2

Proof of correctness

(i) A is a compatible solution as the result of A will always be the product value and not any other abstract value. Also, since the algorithm does not use any repetitions of the array/list values, the output will be compatible.

(ii) A is an optimal set.

Let there be a solution O which is optimal.
This solution has been achieved by removing inversions.
For example

A	B
1	10
2	20
3	30

$\rightarrow 1^{10} * 2^{20} * 2^{10}$

There is an inversion which when solved gives us the optimal solution.
as

$1^{10} * 2^{20} * 3^{30}$.

Our algorithm also produces the same result.

Hence we can say that A is an optimal set.

Question assigned to the following page: 2

(ii) Another way of proving is discussed below

A is an optimal set

(i.e) $\prod a_i^{b_i}$ is the optimal soln.

Let us consider O to be the optimal soln for the last A & B of size 'n'.

Let A' be the solution achieved by my algorithm for the list A & B of size n.

Then by mathematical induction we can say.

→ Base Case : when (n = 1). we know that we only have 1 element in A & B.

By following A', we get :

$$\prod a_1^{b_1} = (\max(A))^{\max(B)}$$

since we sort A & B in descending order in A' This value will always be the greatest value for any A & B.

Hence $\prod(A') \geq \prod(O)$

Question assigned to the following page: 2

Inductive Hypothesis

for $n = k$ where k is a number greater than 1, we get

$\prod_{i=1}^{k} a_i^{b_i}$ = max value for my algorithm A' since we are using the biggest values for base & exponent. This is achieved by getting the max values present in A & B lists. As per our sorting and iteration, we will always get it.

Hence proved for any optimal Seq. O, we will always have the product.

$$\pi(A'(k)) \geq \pi(O(k))$$

Inductive Step

for $(n = n)$ we can now say that we will always have the max base and max exponent of A & B in each iteration with A' algorithm. Hence for any other optimal Seq O, we will always have

$$\pi(A(n)) \geq \pi(O(n))$$

Hence proved.

Question assigned to the following page: 3

3) Greedy Algorithm

Always choose the 2 smallest chain links after every iteration (when you go through all the chains links)

2 5 8 → (2+5) 8 → 7+8
7 15

Total = 2200

Question assigned to the following page: 3

Algo :

We create a min_heap (H) with all weights mentioned in W.

total = 0

while (size(H) != 1)

{

min_wt_first = extract_min(H)

min_wt_second = extract_min(H)

Calories = 100 * (min_wt_first + min_wt_second)

total = total + Calories

insert (min_wt_first + min_wt_second , H)

}

return total ;

Time Complexity

→ Heap construction → O(N)

→ Size of Heap → O(1) performed N times → O(N)

→ Insert to Heap → O(log N) performed N times → O(N log N)

Total = Heap construction + size of heap + 2*insertion.

= O(N) + O(N) + 2*O(N log N)

= O(N log N)

faster than $O(N^2)$