CS3230: Design and Analysis of Algorithms
Semester 2, 2022-23, School of Computing, NUS

Programming Assignment 3

Deadline: 31rd March, 2023 (Friday), 11:59 pm

Instructions

- Use the same account at `https://www.cs3233.com` — by now you know the drill (and this is the last PA).
- Go to `https://www.cs3233.com/contests/` and then click "Register" for "CS3230 PA3 (17-31 Mar 2023)" session (again, please ignore CS3233 related contests, they are not for CS3230). There are 2 tasks (A and B, but both are split into A1-A2 and B1-B2, the usual weaker-harder format). The system will allow submissions from 17 March, 2023 (Friday), at 09:00am onwards but this file will only be uploaded one week later as it contains reflection questions that are a bit too-spoilery.
- Solve the given tasks using any of these 3 programming languages (sorry we do not support any other programming language): C++ (preferred, can solve all tasks comfortably within the time limit if you use the correct algorithm), Python (likely can solve A1 and B1 only, with no guarantee on A2 and B2), or Java (in-between, in terms of runtime speed).
- For every submissions, please put "// TXY_A0123456C" (C++ or Java) or "# TXY_A0123456C" (Python) as the FIRST LINE of any submitted code to `https://www.cs3233.com`, of course after replacing XY with your tutorial group number and A0123456C with your own student number, to aid with the pairwise plagiarism checker later.
- Heavy one-strike-all-CA-marks-gone penalty is applicable to anyone who is proven beyond reasonable doubt submitting code that is 'identical' (with very high percentage of similarities) with another submission(s). Please deviate as far as possible from any AI-generated code as they will be definitely be used as one of the sources for comparison. **Remarks: Steven is incredibly busy during this end March period and may only retrospectively check PA1, PA2, and later PA3 'similar' submissions by mid April; bottom line: be clean**.
- What will be graded is not just your submissions at `https://www.cs3233.com`, but your manual reflection-based answers too (see below).
- After you are done submitting your code at `https://www.cs3233.com`, also submit the soft copy of your reflection-based answers at Canvas → Assignment → Programming Assignment 3.
- You can assume that the web server that hosts `https://www.cs3233.com` can perform up to 10^8 basic operations in 1s — when it is not heavily loaded by multiple submissions at the same time.
- Write legibly. If we cannot read what you write, we cannot give points. In case you CANNOT write legibly, please type out your answers.
- When you submit your answer, please try to make it concise. However, it should contain all the relevant details. The page limit is **now reduced** to just 4 pages. Let's make things shorter this time.
- If you need any clarification on any of the questions, we strongly encourage you to post in Class Discord server (instead of emailing us) so that everybody can see our responses. You may also approach Steven, Diptarka or your tutor.

Task 1 - Min Partition [20 marks]:

13 marks Solve task A1 at https://www.cs3233.com/contest/23/A1/ before deadline.
No further action needed if your solution gets Accepted (after plagiarism check).
To save grading time at the end of the semester (our part-time TAs are incredibly busy too), we turn off TA 'last debugging service', i.e., there is no partial mark this time... instead, see the guiding sub-question hints below.

7 marks Next, answer the following reflection questions about A1 (just a short answer each will be enough).

(a) In Lecture on Week 08, you learn about the existence of the 0/1-Knapsack (call it MAX-KNAPSACK) optimisation problem: Given input $(w_1, v_1), (w_2, v_2), \ldots, (w_N, v_N)$ and W, output a subset $S \subseteq \{1, 2, \ldots, N\}$ that maximizes $\sum_{i \in S} v_i$ such that $\sum_{i \in S} w_i \leq W$.

In this PA3-A1, you are given the MIN-PARTITION optimisation problem: Given input $S = \{s_1, s_2, \ldots, s_N\}$, partition/divide S into two subsets S_1 and S_2 such that $X = sum(S_1)$, $Y = sum(S_2)$, $X \geq Y$, and $X - Y$ is minimized.

In Lecture on Week 10, you learn about polynomial-time reduction.

Which of the two options below is the correct reduction if we want to solve PA3-A1 MIN-PARTITION (a 'new' problem) using what we have learned earlier: MAX-KNAPSACK[1] [1 mark]?

i. We reduce MIN-PARTITION $\leq_p$ MAX-KNAPSACK
ii. We reduce MAX-KNAPSACK $\leq_p$ MIN-PARTITION

(b) Show how to do that polynomial-time reduction! [3 marks]

(c) Then, what do you use to solve the MAX-KNAPSACK problem? Pick one [1 mark]

i. Complete Search (Brute Force) Algorithm
ii. Divide and Conquer (D&C) Algorithm
iii. Dynamic Programming (DP) Algorithm
iv. Greedy Algorithm
v. Randomized Algorithm

(d) What is the time complexity of your solution? Is it a polynomial solution? [1 mark]

(e) Regardless whether you actually solve the harder A2 variant (7x larger N and 21x lesser runtime), which of the following option is best course of action? [1 mark]

i. We can solve A2 using a polynomial algorithm.
ii. We can solve A2 using a greedy algorithm.
iii. We can solve A2 using a randomized algorithm.
iv. None of the above, we must use something else to solve A2.

Task 2 - Recombination [20 marks]:

7 marks Solve task B1 at https://www.cs3233.com/contest/23/B1/ before deadline.
No further action needed if your solution gets Accepted (after plagiarism check).
Again, no partial marks to save grading time.

6 marks Solve task B2 at https://www.cs3233.com/contest/23/B2/ before deadline.
No further action needed if your solution gets Accepted (after plagiarism check).
Again, no partial marks to save grading time.

[1] FAQ: If you use the 'simpler' SUBSET-SUM problem (that is not directly discussed in CS3230) in your actual solution (or any other alternative solutions), please still answer these reflection questions assuming that you use what you had learned on Week 08: MAX-KNAPSACK.

7 marks Next, answer the following reflection questions about B1+B2 (just a short answer each will be enough).

(a) Let's use polynomial-time reduction again. This 'new' problem: RECOMBINATION is similar to another problem that you have also learned in Week 08: LONGEST-COMMON-SUBSEQUENCE, or LCS. Show how to get the minimum number of minutes that Okabe needs to spend to spend to one of the M virus (let's say virus O) into his favorite one (let's say virus F) in terms of LCS. Both O and F will be of length N. There is a simple reduction for this [1 mark]. We will then repeat this M times.

i. $LCS(F, O)$
ii. $N + LCS(F, O)$
iii. $min(N, LCS(F, O))$
iv. $max(N, LCS(F, O))$
v. None of the above, the correct reduction is... (elaborate).

(b) What did you learn back in Week 08 to compute the LCS of two arrays (we can treat a string — we learned this one in Week 08 – as a character array)? It was one of these options: [1 mark]

i. Complete Search (Brute Force) Algorithm
ii. Dynamic Programming (DP) Algorithm
iii. Randomized Algorithm

(c) What is the time complexity of that solution (of the previous sub-question)?
Remember that Okabe has M viruses and there are TC test cases in total.
Explain the time complexity in terms of TC, M, and N!
Can this $O(\cdot)$ time complexity passes B2 constraints? [1 mark]

(d) There is an 'optional' part of Week 09 lecture about Greedy algorithms that because of this PA3-B2, is no longer optional. It is about another problem called LONGEST-INCREASING-SUBSEQUENCE (LIS). And when we look at the viruses that Okabe has, all of them are 'permutations of length N'. Can you use this information to further do polynomial time reduction from RECOMBINATION $\leq_p$ LCS $\leq_p$ LIS? [2 marks].

(e) The last jigsaw piece after we reduce RECOMBINATION $\leq_p$ LCS $\leq_p$ LIS is on how to compute LIS faster than $O(N^2)$ – otherwise we are back to square one. For this, we need to properly learn 'Patience Solitaire' algorithm (the optional recording at Canvas → Videos/Panopto → Optional Material → Week-9-LIS using Greedy (Optional Video)). Summarize this algorithm and explain full solution in your own words (perhaps Chat-GPT assisted) and analyze its overall $O(\cdot)$ time complexity, again in terms of TC, M, and N. [2 marks]

Task 1 - Min Partition [20 marks]:

13 marks Solve task A1 at https://www.cs3233.com/contest/23/A1/ before deadline.
No further action needed if your solution gets Accepted (after plagiarism check).
To save grading time at the end of the semester (our part-time TAs are incredibly busy too), we turn off TA 'last debugging service', i.e., there is no partial mark this time... instead, see the guiding sub-question hints below.

7 marks Next, answer the following reflection questions about A1 (just a short answer each will be enough).

(a) In Lecture on Week 08, you learn about the existence of the 0/1-Knapsack (call it MAX-KNAPSACK) optimisation problem: Given input $(w_1, v_1), (w_2, v_2), \ldots, (w_N, v_N)$ and W, output a subset $S \subseteq \{1, 2, \ldots, N\}$ that maximizes $\sum_{i \in S} v_i$ such that $\sum_{i \in S} w_i \leq W$.

In this PA3-A1, you are given the MIN-PARTITION optimisation problem: Given input $S = \{s_1, s_2, \ldots, s_N\}$, partition/divide S into two subsets S_1 and S_2 such that $X = sum(S_1)$, $Y = sum(S_2)$, $X \geq Y$, and $X - Y$ is minimized.

In Lecture on Week 10, you learn about polynomial-time reduction.

Which of the two options below is the correct reduction if we want to solve PA3-A1 MIN-PARTITION (a 'new' problem) using what we have learned earlier: MAX-KNAPSACK[1] [1 mark]?

i. We reduce MIN-PARTITION $\leq_p$ MAX-KNAPSACK
ii. We reduce MAX-KNAPSACK $\leq_p$ MIN-PARTITION

(b) Show how to do that polynomial-time reduction! [3 marks]

(c) Then, what do you use to solve the MAX-KNAPSACK problem? Pick one [1 mark]

i. Complete Search (Brute Force) Algorithm
ii. Divide and Conquer (D&C) Algorithm
iii. Dynamic Programming (DP) Algorithm
iv. Greedy Algorithm
v. Randomized Algorithm

(d) What is the time complexity of your solution? Is it a polynomial solution? [1 mark]

(e) Regardless whether you actually solve the harder A2 variant (7x larger N and 21x lesser runtime), which of the following option is best course of action? [1 mark]

i. We can solve A2 using a polynomial algorithm.
ii. We can solve A2 using a greedy algorithm.
iii. We can solve A2 using a randomized algorithm.
iv. None of the above, we must use something else to solve A2.

a) ii

b) We must make the partition problem into a knapsack problem by viewing the bag as the sum of all elements divided by 2

c) iii

d) $O(n^2)$ since there is a nested for loop

e) iv

7 marks Next, answer the following reflection questions about B1+B2 (just a short answer each will be enough).

(a) Let's use polynomial-time reduction again. This 'new' problem: RECOMBINATION is similar to another problem that you have also learned in Week 08: LONGEST-COMMON-SUBSEQUENCE, or LCS. Show how to get the minimum number of minutes that Okabe needs to spend to spend to one of the M virus (let's say virus O) into his favorite one (let's say virus F) in terms of LCS. Both O and F will be of length N. There is a simple reduction for this [1 mark]. We will then repeat this M times.

i. $LCS(F, O)$
ii. $N + LCS(F, O)$
iii. $min(N, LCS(F, O))$
iv. $max(N, LCS(F, O))$
v. None of the above, the correct reduction is... (elaborate).

(b) What did you learn back in Week 08 to compute the LCS of two arrays (we can treat a string — we learned this one in Week 08 – as a character array)? It was one of these options: [1 mark]

i. Complete Search (Brute Force) Algorithm
ii. Dynamic Programming (DP) Algorithm
iii. Randomized Algorithm

(c) What is the time complexity of that solution (of the previous sub-question)?
Remember that Okabe has M viruses and there are TC test cases in total.
Explain the time complexity in terms of TC, M, and N!
Can this $O(\cdot)$ time complexity passes B2 constraints? [1 mark]

(d) There is an 'optional' part of Week 09 lecture about Greedy algorithms that because of this PA3-B2, is no longer optional. It is about another problem called LONGEST-INCREASING-SUBSEQUENCE (LIS). And when we look at the viruses that Okabe has, all of them are 'permutations of length N'. Can you use this information to further do polynomial time reduction from RECOMBINATION $\leq_p$ LCS $\leq_p$ LIS? [2 marks].

(e) The last jigsaw piece after we reduce RECOMBINATION $\leq_p$ LCS $\leq_p$ LIS is on how to compute LIS faster than $O(N^2)$ – otherwise we are back to square one. For this, we need to properly learn 'Patience Solitaire' algorithm (the optional recording at Canvas → Videos/Panopto → Optional Material → Week-9-LIS using Greedy (Optional Video)). Summarize this algorithm and explain full solution in your own words (perhaps Chat-GPT assisted) and analyze its overall $O(\cdot)$ time complexity, again in terms of TC, M, and N. [2 marks]

a) v as finding the difference between n and LCS provides our answer

b) ii

c) $O(n^2 \cdot m \cdot TC)$ since finding the LCS is n^2 and we do that m times for each virus amongst TC cases

d) Yes, we use LIS to reduce the recombination for B2

e)

The algorithm maintains a list of piles represented by its top card. We iterate over the numbers and perform a binary search on each pile. We add the number to the first pile $\geq$ current number else create a new pile.

Time complexity is $O(TC * M * N * \log(N))$.