COP4533 - Programming Assignment
Milestone-1 Report

This document provides a template for writing the report for MILESTONE-1. Its purpose is to give you a clear base structure for your write-up, along with tips and suggestions on what to include in each section. To illustrate the expected level of detail, the template uses the *Interval Scheduling* problem as an example.

1 Team Members

If you are working as a team, list the names of all team members and briefly describe each member's contribution.

2 Algorithm Design and Analysis

2.1 Algorithm 1 (ProblemS1)

2.1.1 Description

Describe your greedy algorithm for ProblemS1. Explain how it works step by step. Include pseudocode if appropriate.

The purpose of the algorithm description is to communicate your approach as effectively as possible. A long narrative describing each line of your code is not ideal. Instead, aim for a concise, high-level summary of your algorithm, avoiding unnecessary implementation details. Including pseudocode can help clarify your approach, but keep it brief. It's best to include only language-independent, relevant details. For example, *make_tuple* is specific to C++ and unnecessary for algorithm explanation. Strive for readability; good pseudocode balances clarity and conciseness while covering essential details. Try comparing with pseudo codes you come across in books. Here's a helpful resource on writing effective pseudocode: `https://www.cs.cornell.edu/courses/cs482/2003su/handouts/pseudocode.pdf`

Example using interval scheduling problem:

We can solve the interval scheduling problem using the following greedy strategy.

Greedy strategy: Always choose the activity that finishes earliest among those compatible with the already chosen activities.

Step by step example: Suppose we have the intervals (start, finish):

$$(1,4),(3,5),(0,6),(5,7),(3,9),(5,9),(6,10),(8,11),(8,12),(2,14),(12,16).$$

1. Select $(1,4)$, the earliest finishing interval.
2. The next compatible interval is $(5,7)$.
3. Then choose $(8,11)$.
4. Finally, choose $(12,16)$.

The resulting set is $\{(1,4),(5,7),(8,11),(12,16)\}$ with 4 activities.

Algorithm 1 Interval Scheduling (Earliest Finish Time First)

```
1: procedure IntervalScheduling($n, (s_1, f_1), (s_2, f_2), \ldots, (s_n, f_n)$)
2:     Sort all jobs by finish times so that $f_1 \le f_2 \le \cdots \le f_n$
3:     $S \leftarrow \emptyset$                                   ▷ set of jobs selected
4:     for $j = 1$ to $n$ do
5:         if job $j$ is compatible with $S$ then
6:             $S \leftarrow S \cup \{j\}$
7:         end if
8:     end for
9:     return $S$
10: end procedure
```

2.1.2 Correctness Proof

Provide a clear argument or formal proof that your algorithm always gives the correct solution for ProblemS1.

Correctness proofs are typically the weakest part of most submissions. When reviewing your proof, ask yourself if it could "prove" an incorrect algorithm.

Example using interval scheduling problem:

Theorem.

The earliest-finish-time-first greedy algorithm produces an optimal solution to the interval scheduling problem.

Proof. [by contradiction]

Assume that the greedy algorithm is not optimal.

1. Let $g_1, g_2, \ldots, g_k$ denote the set of jobs selected by the greedy algorithm.

2. Let $o_1, o_2, \ldots, o_m$ denote the set of jobs in an optimal solution with maximal r such that $o_i = g_i$ for all $i \leq r$.

3. Consider the job at position $r+1$. By construction, the greedy choice g_{r+1} is the earliest finishing job compatible with $g_1, g_2, \ldots, g_r$ and $o_1, o_2, \ldots, o_r$. Hence $finish\_time(g_{r+1}) \leq finish\_time(o_{r+1})$.

4. Replace o_{r+1} with g_{r+1} to form a new solution. This replacement preserves feasibility because g_{r+1} finishes no later than o_{r+1} and is compatible with $o_1, \ldots, o_r$.

5. The new solution has size at least as large as the optimal solution O, but it agrees with the greedy solution on $r+1$ jobs, contradicting the maximality of r.

Hence, the greedy algorithm is optimal.

2.1.3 Runtime Analysis

Analyze the time complexity of Algorithm 1. Make sure to include sufficient justification.

Example using interval scheduling problem:

Let n be the number of jobs. The algorithm begins by sorting the jobs by their finish times, which requires $O(n \log n)$ time. After sorting, the algorithm scans through the list once, adding each compatible job to the solution. This step takes $O(n)$ time.

Therefore, the overall running time is

$$O(n \log n) + O(n) = O(n \log n).$$

Since the sorting step dominates, the greedy interval scheduling algorithm runs in $O(n \log n)$ time.

2.2 Question 1

Provide an input example showing that Algorithm 1 does not always solve ProblemG. Explain why it fails on this input.

2.3 Question 2

Provide an input example showing that Algorithm 1 does not always solve ProblemS2. Explain why it fails on this input.

2.4 Algorithm 2 (ProblemS2)

2.4.1 Description

Describe your greedy algorithm for ProblemS2. Explain how it works step by step. Include pseudocode if appropriate.

2.4.2 Correctness Proof

Provide a clear argument or formal proof that your algorithm always gives the correct solution for ProblemS2.

2.4.3 Runtime Analysis

Analyze the time complexity of Algorithm 2. Make sure to include sufficient justification.

3 Experimental Comparative Study

The goal of the experimental study is to visualize the growth of the running time of the algorithms as the input size increases. Few things that will make the visualization more accurate.

- Generate datasets with size n that goes high enough. Small values of n leads to an inaccurate visualization.
- Pick uniformly distributed values of n. Non uniformly distributed values of n leads to a misleading visualization.
- Use enough samples. Not enough samples of n also leads to an inaccurate visualization.

3.1 Experimental Setup

Describe how you generated random datasets, their sizes, and how you measured running times.

You may include additional plots

3.2 Plot 1

Insert your plot of running time for Algorithm 1 vs. input size. Explain the observed trend.

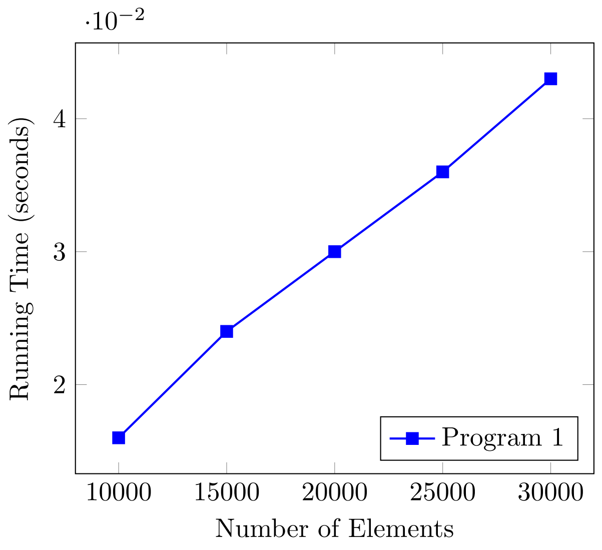

Figure 1: Example plot.

3.3 Plot 2

Insert your plot of running time for Algorithm 2 vs. input size. Explain the observed trend.

3.4 Observations/Comments

Inlcude any additional observations or comments.

4 Conclusion

Summarize your learning experience in Milestone 1. Reflect on design, analysis, experiments, and challenges.