ShanghaiTech University

CS240 Algorithm Design and Analysis

Fall 2024

Problem Set 1

Due: 23:59, Mar. 18, 2025

1. Submit your solutions to the course Gradescope.
2. If you want to submit a handwritten version, scan it clearly.
3. Late submissions are not allowed.
4. You are required to follow ShanghaiTech's academic honesty policies. You are allowed to discuss problems with other students, but you must write up your solutions by yourselves. You are not allowed to copy materials from other students or from online or published resources. Violating academic honesty can result in serious penalties.

Problem 1:

Sort the following functions in ascending order of growth.

$$f_1(n) = 2^{\log_3 n} \tag{1}$$
$$f_2(n) = \log_3 n \tag{2}$$
$$f_3(n) = n^{2025} \tag{3}$$
$$f_4(n) = \log_2 n^n \tag{4}$$
$$f_5(n) = \log_5 2^n \tag{5}$$
$$f_6(n) = n^{\sqrt{\frac{1}{2}}} \tag{6}$$
$$f_7(n) = n^{\log_2 n} \tag{7}$$

Solution:

(2) < (1) < (6) < (5) < (4) < (3) < (7)

Problem 2:

Analyze the time complexities of the following algorithms and explain your reasoning.

Algorithm 1

```
for i ← 1 to n by i ← i + 1 do
    for j ← 0 to (3i)^2 by j ← j + 1 do
        res_1 ← res_1 + i + j
    end for
    for k ← 0 to 2i by k ← k + 1 do
        res_2 ← res_2 + i + k
    end for
end for
```

Solution:

For res_1, the outer loop is executed n times, the inner loop is executed $(3i)^2$ times. $\Sigma_{i=0}^{n}(3i)^2 = 9 \times \Sigma_{i=0}^{n} i^2 = 9 \times n \times (n+1) \times (2 \times n + 1)/6$, which is $O(n^3)$. For res_2, the outer loop is executed n times, the inner loop is executed $2i$ times. $\Sigma_{i=0}^{n} 2i = 2 \times n \times (n+1)/2$, which is $O(n^2)$. The overall complexity is $O(n^3)$.

Problem 3:

You have a magical forecasting model that can predict the future daily earnings of a certain stock, and you plan to use this model to make a fortune. To avoid arousing suspicion, you need to minimize the number of transactions, meaning you should only buy once and sell once every N days. Assuming you already know the daily earnings of this stock for the next N days (which may be positive or negative), please design a divide and conquer algorithm (with complexity analysis) to determine which day to buy and sell in order to maximize your earnings.

Example:

- **Input:** [-2, 1, -3, 4, -1, 2, 1, -5, 4].
- **Output:** 6.
- **Explanation:** Buy the stock on day 4 and sell it on day 7, the earnings are $4 - 1 + 2 + 1 = 6$.

Solution:

This problem requires finding a continuous interval with the largest sum of elements. We use a divide-and-conquer algorithm.

Divide step: We divide the set of N earnings into two roughly equal halves. Recursively find the continuous interval with the largest sum for both the left and right halves.

Combine step: For the left and right halves, find the maximum continuous sequence sum across the two parts. Compare it with the maximum continuous sequence sum of the left and right halves, and take the largest value as the maximum value of the current interval.

Time complexity: The time complexity of combine step is $O(n)$ for sequence length of n, and each subproblem of left and right halves requires $T(n/2)$. The recurrence relation is: $T(n) = 2T(n/2) + O(n)$. Thus the overall time complexity is $O(n \log n)$.

Problem 4:

After a catastrophe known as Death Stranding, the connections between cities around the world were severed. As a deliveryman, your task is to reconnect these cities to the Cairo network by delivering signal receivers to the respective cities. Since the distances between cities are vast and perilous, you must find the shortest path from the capital city that allows you to traverse all the cities. It is important to note that there is no additional cost for retracing the same path.

Detailed Problem Description: Suppose we have 3 cities, and the distance between the cities can be expressed as a matrix:

$$D = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 3 \\ 2 & 3 & 0 \end{bmatrix},$$

where $D_{i,j}$ represent the distance between the city i and city j. You need to find the roads with the lowest sum of distance that can connect all the cities. In this case, it is $D_{1,2} + D_{1,3} = 3$. You need to provide the algorithm to solve this problem and prove the correctness.

Solution:

We use the Greedy algorithm to achieve the target.

Greedy Algorithm:

- Step 1: Sort all the roads by distance in ascending order.
- Step 2: Choose the shortest road.
- Step 3: Check if the endpoints of the road are already connected, if not, choose this road.
- Step 4: Repeat the step 2,3 until all the cities are connected.

Proof: Assume there exists an optimal solution with the set of roads S_{opt} that has a strictly shorter distance than the greedy algorithm S_{greedy}. We represent the problem as a graph G, with V as set of cities, E as set of roads between cities. Let e' be the first road in S_{greedy} not in S_{opt}. When e' was added by the algorithm, it connected two components C_1 and C_2, where $C_1, C_2 \subset V$. S_{opt} must contain a path P connecting C_1 and C_2, which includes a road e crossing the cut (C_1, C_2). Since the algorithm chooses the smallest edge across this cut, $d(e') \leq d(e)$, where $d()$ is the distance of connections.

Replace e with e' in S_{opt}. The new set $S_{\text{new}} = S_{\text{opt}} \setminus \{e\} \cup \{e'\}$ is still connecting all cities, and its total distance satisfies:

$$d(S_{\text{new}}) = d(S_{\text{opt}}) - d(e) + d(e') \leq d(S_{\text{opt}}).$$

This contradicts the assumption that S_{opt} is strictly better than S_{greedy}. Hence, S_{greedy} must be optimal.