Weekly Assignment 5
Total: 100

CS 2500: Algorithms

Due Date: November 18, 2024 at 11.59 PM

Instructions

- Submit your solutions by the deadline specified above.
- Ensure that your work is your own.
- Write your answers clearly and show all your work.
- If you have any questions, ask during recitations or office hours.

Problems

1. You are given a set of four jobs, each with a processing time t_j. Your goal is to find an optimal order to schedule the jobs that minimizes the total turnaround time (sum of waiting times plus processing times for all jobs).

   Given:

   - Job A: $t_A = 3$ units
   - Job B: $t_B = 8$ units
   - Job C: $t_C = 2$ units
   - Job D: $t_D = 5$ units

   (a) Determine the turnaround time for each possible scheduling order. **[3]**

   (b) Identify the order that results in the minimum average turnaround time. **[4]**

   (c) Explain why scheduling the jobs in ascending order of processing time minimizes the average turnaround time. **[3]**

2. You are given a set of five jobs, each with a profit p_j and a deadline d_j. Your goal is to schedule the jobs to maximize the total profit, ensuring that each job can be completed before its deadline.

   Given:

   - Job 1: $p_1 = 10$, $d_1 = 1$
   - Job 2: $p_2 = 75$, $d_2 = 3$
   - Job 3: $p_3 = 15$, $d_3 = 2$
   - Job 4: $p_4 = 40$, $d_4 = 4$

   (a) Determine a job sequence that maximizes the total profit while ensuring each job is completed before its deadline. **[4]**

   (b) For the sequence you created, calculate the total profit. **[4]**

(c) Explain why the greedy approach of selecting jobs in non-increasing order of profit and checking feasibility before each addition is effective. **[2]**

3. The English coinage before decimalization included half-crowns (30 pence), florins (24 pence), shillings (12 pence), sixpences (6 pence), threepences (3 pence), and pennies (not to mention ha'pennies and farthings, worth respectively $\frac{1}{2}$ and $\frac{1}{4}$ pence). Show that with these coins the greedy algorithm `MakeChange` on Slide 8 of Lecture 19 does not necessarily produce an optimal solution, even when an unlimited supply of coins of each denomination is available. **[8]**

4. What can you say about the time required by Kruskal's algorithm if, instead of providing a list of edges, the user supplies a matrix of distances, leaving to the algorithm the job of working out which edges exist? **[6]**

5. Suppose Kruskal's algorithm and Prim's algorithm are implemented as shown in Slide 21 and Slide 32 respectively of Lecture 21. What happens (a) in the case of Kruskal's algorithm (b) in the case of Prim's algorithm if by mistake we run the algorithm on a graph that is not connected? **[6]**

6. A graph may have several different minimum spanning trees. Is this the case for the graph on Slide 11 Lecture 21? If so, where is this possibility reflected in the algorithms explained Slide 21 and Slide 32 respectively? **[5]**

7. In Dijkstra's algorithm, when we add a new node v to the set S of `PERM` nodes, let w be a node not in S. Is it possible that the new shortest path from the source to w should first pass through v and then by some other node of S? **[6]**

8. Show by giving an explicit example that if the edge lengths can be negative, then Dijkstra's algorithm does not always work correctly. Is it still sensible to talk about shortest paths if negative distances are allowed? **[8]**

9. A knapsack has a maximum weight capacity of 100 units. There are five objects available, each with a specific weight and value, as given in the table below:

Object	Weight (W)	Value (V)
1	10	20
2	20	30
3	30	66
4	40	40
5	50	60

Using the fractional knapsack approach:

(a) Calculate the value-to-weight ratio for each object. **[4]**

(b) Determine the optimal selection of items (including fractions if necessary) to maximize the total profit, without exceeding the knapsack's weight limit of 100 units. **[4]**

(c) Show your calculations and clearly state the maximum profit that can be obtained. **[2]**

10. Consider five characters A, B, C, D, and E with the following probabilities:

Character	Probability
A	0.35
B	0.1
C	0.2
D	0.2
E	0.15

(a) Construct a Huffman code for this set of symbols. **[6]**

(b) Encode the text `ABCAD` using the generated code. **[4]**

11. Use Kruskal's algorithm to construct a Minimum Spanning Tree (MST) for the following graph: **[5]**

12. Apply Dijkstra's algorithm for finding the shortest path using node A as source node: **[6]**

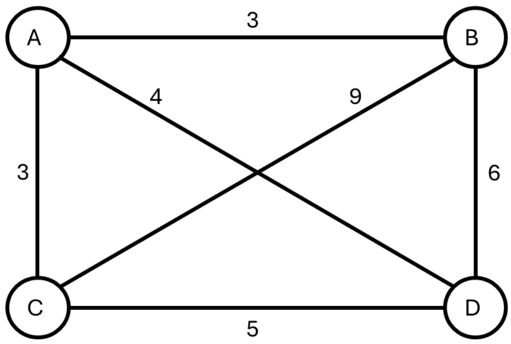

Figure 1: Example graph for MST

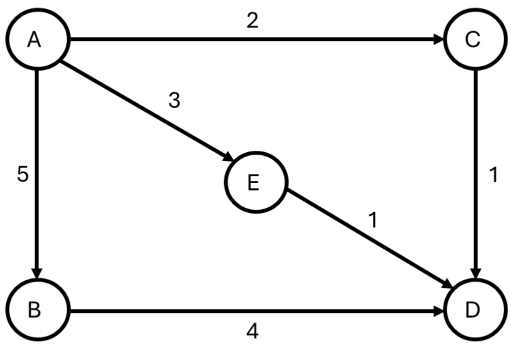

Figure 2: Example graph for Dijkstra's Algorithm

Coding

Implement Dijkstra's Algorithm

The zip file contains a file named `dijkstra.py`, which provides a foundational structure for implementing Dijkstra's Algorithm. This starter code includes placeholders marked as `TODO`, guiding you to complete specific parts of the algorithm. Follow the instructions in these TODO sections to implement a fully functional version of Dijkstra's Algorithm. [**10**]

Extra Credit: Implement Heap Sort

The zip file contains a file named `heapsort.py`, which provides a foundational structure for implementing the Heap Sort algorithm in Python. This starter code includes placeholders marked as `TODO`, guiding you to complete specific parts of the algorithm. Complete the code in `heapsort.py` by filling in the `TODO` sections. Once completed, apply your Heap Sort implementation to the array presented on Slide 31 of Lecture 12. Ensure that you show both the initial array and the sorted array as output. [**10**]

Note: We provide the starter code in Python only. If you choose to use another programming language, you will need to adapt the code on your own.