CPSC 320 2023S2: Assignment 4 Solutions

3 More Recurrences

Consider this recurrence relation:

$$T(n) = \begin{cases} 1, & \text{for } n < n_0, \\ T(n/2) + T(5n/9) + T(4n/7) + n^2, & \text{for } n \geq n_0, \end{cases}$$

Unfortunately, the Master Theorem doesn't apply directly to this recurrence, and expanding out the recursion tree looks very painful! However, there are still ways to solve (or at least bound) this recurrence!

1. (1 mark) Consider this nicer recurrence relation:

$$L(n) = \begin{cases} 1, & \text{for } n < n_0, \\ L(n/2) + L(n/2) + L(n/2) + n^2, & \text{for } n \geq n_0, \end{cases}$$

   Fortunately, the Master Theorem does apply to this recurrence! What are the values of a, b, and k, and what does the Master Theorem give as the big-Θ bound for $L(n)$?

   SOLUTION: $a = 3$, $b = 2$, and $k = 2$. Using the Worksheet version of the Master Theorem, we find that $a = 3 < 4 = 2^2 = b^k$. Therefore, $L(n) = \Theta(n^2)$.

2. (3 marks) Prove that $L(n) \leq T(n)$ for all $n > 0$. You may assume that n_0 is whatever is convenient so that your base cases work out. You may also assume that both $L(n)$ and $T(n)$ are monotonically non-decreasing, e.g., for all $m < n$, you have $L(m) \leq L(n)$ and $T(m) \leq T(n)$. Hint: This should be a very short strong induction proof. Here and for the rest of the problem you can ignore floors and ceilings, i.e., assume that when you divide n by a constant the result is an integer.

   SOLUTION: In the base case, $n < n_0$, and $L(n) = 1 = T(n)$, so $L(n) \leq T(n)$.

   In the inductive case, assume that for all $n' < n$, $L(n') \leq T(n')$. We find that:

$$\begin{aligned} L(n) &= L(n/2) + L(n/2) + L(n/2) + n^2 \\ &\leq L(n/2) + L(5n/9) + L(4n/7) + n^2 \quad \text{by monotonicity} \\ &\leq T(n/2) + T(5n/9) + T(4n/7) + n^2 \quad \text{by I.H.} \\ &= T(n). \end{aligned}$$

3. (2 marks) Similarly, define a recurrence relation $U(n)$ such that $T(n) \leq U(n)$ for all $n > 0$, and where you can directly solve $U(n)$ via the Master Theorem. You should try to make $U(n)$ as tight to $T(n)$ as possible.

   SOLUTION:

$$U(n) = \begin{cases} 1, & \text{for } n < n_0, \\ U(4n/7) + U(4n/7) + U(4n/7) + n^2, & \text{for } n \geq n_0, \end{cases}$$

   We relied on the fact that $4/7 > 5/9 > 1/2$.

4. (2 marks) Prove that $T(n) \le U(n)$ for all $n > 0$. (Your proof should be very similar to the proof you did for $L(n)$.)

   SOLUTION: To prove that $T(n) \le U(n)$, we do another strong induction:

   In the base case, $n < n_0$, and $T(n) = 1 = U(n)$, so $T(n) \le U(n)$.

   In the inductive case, assume that for all $n' < n$, $T(n') \le U(n')$. We find that:

$$\begin{aligned} T(n) &= T(n/2) + T(5n/9) + T(4n/7) + n^2 \\ &\le T(4n/7) + T(4n/7) + T(4n/7) + n^2 \quad \text{by monotonicity} \\ &\le U(4n/7) + U(4n/7) + U(4n/7) + n^2 \quad \text{by I.H.} \\ &= U(n). \end{aligned}$$

5. (1 mark) What are the values for a, b, and k for your $U(n)$, and what is the resulting big-Θ bound for $U(n)$ that the Master Theorem gives you? (Hint: You'll probably want a calculator to do some arithmetic with 1/2, 5/9, and 4/7, their reciprocals, and their squares...)

   SOLUTION: $a = 3$, $b = 7/4$, and $k = 2$. Using the Worksheet version of the Master Theorem, we find that $a = 3 < 3.0625 = (7/4)^2 = b^k$. Therefore, $U(n) = \Theta(n^2)$.

6. (1 mark) What big-Θ bound can you conclude for $T(n)$?

   SOLUTION: Since $L(n) \le T(n) \le U(n)$, and both $L(n) = \Theta(n^2)$ and $U(n) = \Theta(n^2)$, we know that $T(n) = \Theta(n^2)$, too.

4 Travel Itinerary

You are the manager of the Abbotsford branch of a large multinational company, and your company just bought out a small business based in Brampton, Ontario. However, your company has not found a manager for the Brampton branch yet, so they have asked you to oversee both branches until the end of the year. There are n days remaining in the year.

Because you will be spending a fair amount of time in Brampton, you have determined that it will be more convenient to stay in hotels in both cities, rather than continuing to rent your apartment in Abbotsford. You cannot stay in either location for more than 7 days because you do not want to be away from either branch for too long. You must determine how to split your time between Abbotsford and Brampton in a way that minimizes the total cost of your hotels and flights.

The cost of your flight/hotel depends on the day (weekends/holidays might be more expensive, or there might be promotions on certain days) and the city. You are given four arrays that provide the cost of the hotels and flights for each day.

The arrays f_A, f_B, h_A, and h_B, each of length n, are defined as follows:

> $f_A[i]$ is the cost of flying *to* Abbotsford on day i
>
> $f_B[i]$ is the cost of flying *to* Brampton on day i
>
> $h_A[i]$ is the cost of staying in a hotel in Abbotsford on day i
>
> $h_B[i]$ is the cost of staying in a hotel in Brampton on day i

You are looking to minimize the cost of accommodations plus flights, subject to the constraint that you cannot spend more than 7 days in one location. You can assume that your flights are always in the morning (so if you are flying between cities on day i, you only have to pay for a hotel in your destination city that day).

Let $C_A[i]$ be the lowest possible value of the objective function for days 1 to i, assuming you are staying in Abbotsford on day i. Similarly, let $C_B[i]$ be the lowest possible value of the objective function for days 1 to i, assuming you are staying in Brampton on day i. In either case, assume that you are staying in Abbotsford on day 0. The solutions to Tutorial 7 provide recurrences for these values.

1. (4 marks) Using the recurrences from Tutorial 8, write pseudo-code for an iterative dynamic programming algorithm that takes f_A, f_B, h_A and h_B as input and populates the arrays C_A and C_B for every value $1 \leq i \leq n$. Note: it is acceptable to leave min() operations in your pseudo-code instead of writing out an inner loop.

   SOLUTION:

Algorithm 1 Iteratively populate C_A and C_B.

```
1: procedure OptimalCost($h_A$, $h_B$, $f_A$, $f_B$)
2:     create 1D arrays $C_A[0..n]$ and $C_B[0..n]$
3:     $C_A[0] \leftarrow 0$
4:     $C_A[1] \leftarrow h_A[(1)$
5:     $C_B[0] \leftarrow 0$
6:     $C_B[1] \leftarrow f_B(1) + h_B(1)$
7:     for $i \leftarrow 2$ to $7$ do
8:         $C_A[i] = \min\left(\min_{1\le x<i}\left(C_B[i-x] + f_A[i-x+1] + \sum_{j=i-x+1}^{i} h_A(j)\right),\ \ \sum_{j=1}^{i} h_A(j)\right)$
9:         $C_B[i] = \min_{1\le x\le i}\left(C_A[i-x] + f_B[i-x+1] + \sum_{j=i-x+1}^{i} h_B(j)\right)$
10:    for $i \leftarrow 8$ to $n$ do
11:        $C_A[i] = \min_{1\le x\le 7}\left(C_B[i-x] + f_A[i-x+1] + \sum_{j=i-x+1}^{i} h_A(j)\right)$
12:        $C_B[i] = \min_{1\le x\le 7}\left(C_A[i-x] + f_B[i-x+1] + \sum_{j=i-x+1}^{i} h_B(j)\right)$
```

2. (1 mark) State the running time of your algorithm from part 1 as a function of n. No justification needed.

   SOLUTION:

   The algorithm runs in $\Theta(n)$ time.

5 Max-Weight Independent Sets

An *independent set* of an undirected graph G is a subset of nodes, no two of which is connected by an edge. This problem concerns independent sets on "rail line" graphs. These undirected graphs have $2n$ nodes, labeled $[1, i]$, $[2, i]$, $1 \le i \le n$. A function $w()$ assigns a weight to each node. There are edges between $[1, i]$ and $[2, i]$ for each i, as well as edges between $[1, i]$ and $[1, i+1]$ for $1 \le i \le n-1$ and between $[2, i]$ and $[2, i+1]$ for $1 \le i \le n-1$, for a total of $2(n-1)+n = 3n-2$ edges.

For a subset S of the nodes of G, let $w(S)$ be the sum of the weights of the nodes of S. We want to find the max possible weight of an independent set of G. That is, given G and the weight function $w()$, we want to compute

$$\text{WMIS}(G) = \max_{S | S \text{ is an independent set of G}} w(S)$$

We'll describe WMIS(G) in terms of three quantities. These describe the weight of the max-weight independent set of G containing only nodes labeled $[1, j]$ or $[2, j]$, $1 \le j \le i$, but with different constraints:

- $\text{WMIS}^{+-}(G, i)$: the set must include $[1, i]$ but exclude $[2, i]$.
- $\text{WMIS}^{-+}(G, i)$: the set must include $[2, i]$ but exclude $[1, i]$.
- $\text{WMIS}^{--}(G, i)$: the set must exclude both $[2, i]$ and $[1, i]$.

(We do not include WMIS^{++} because an independent set can't contain both $[1, i]$ and $[2, i]$, since there is an edge between them.)

1. (1 mark) Explain why

$$\text{WMIS}(G) = \max\{\text{WMIS}^{+-}(G, n), \text{WMIS}^{-+}(G, n), \text{WMIS}^{--}(G, n)\}.$$

SOLUTION: This is because the three terms within the max expression correspond to the three possibilities for the max-weight independent set for G: it contains $[1, n]$ but not $[2, n]$; it contains $[2, n]$ but not $[1, n]$; or it contains neither $[1, n]$ nor $[2, n]$.

2. (2 marks) Explain why

$$\text{WMIS}^{+-}(G, i) = \begin{cases} 0, & \text{when } i = 0, \\ w([1, i]) + \max\{\text{WMIS}^{-+}(G, i-1), \text{WMIS}^{--}(G, i-1)\}, & \text{when } i \ge 1. \end{cases}$$

SOLUTION: In base case, when $i = 0$, there can be no nodes in the independent set and the weight of the empty set is 0.

When $i \ge 1$, the expression on the right hand side of the recurrence accounts for the weight of the included node $[1, i]$, as well as the max weight of the remaining set, which cannot include node $[1, i-1]$ since it has an edge to $[1, i]$. The node $[2, i-1]$ may or may not be included, so we take the max of these two possibilities.

3. (2 marks) Write down a recurrence for $\text{WMIS}^{-+}(G, i)$. No justification needed.

SOLUTION:

$$\text{WMIS}^{-+}(G, i) = w([2, i]) + \max\{\text{WMIS}^{+-}(G, i-1), \text{WMIS}^{--}(G, i-1)\}.$$

4. (2 marks) Write down a recurrence for $\text{WMIS}^{--}(G, i)$. No justification needed.

SOLUTION:

$$\text{WMIS}^{--}(G, i) = \max\{\text{WMIS}^{+-}(G, i-1), \text{WMIS}^{-+}(G, i-1), \text{WMIS}^{--}(G, i-1)\}.$$