محمد اسامه محمد حلمي عبد الهادي

Step 1: Recall Master Theorem

The general recurrence for the Master Theorem is:

T(n)=aT(nb)+f(n),T(n) = aT\left(\frac{n}{b}\right) + f(n),T(n)=aT(bn)+f(n),

where:

- aaa: the number of subproblems,
- bbb: the factor by which the input size is divided,
- f(n)f(n)f(n): the cost of work done outside the recursive calls.

We analyze the recurrence in terms of npn^pnp, where: p=log b a

Step 2: Identify parameters

For the given recurrence:

- a=4a = 4a=4,
- b=2b = 2b=2,
- f(n)=n3f(n) = n^3f(n)=n3

Compute p=log b a

p= log 2 4 =2

Step 3: Compare f(n)=n3f(n) = n^3f(n)=n3 with np=n2n^p = n^2np=n2

- f(n)=n3f(n) = n^3f(n)=n3 grows faster than n2n^2n2, i.e., f(n)∈ω(np)f(n) \in \omega(n^p)f(n)∈ω(np).
- Now, we check the **regularity condition**: f(n)f(n)f(n) must dominate npn^pnp by at least a polynomial factor. Specifically, we check if: f(n)np=n3n2=n,\frac{f(n)}{n^p} = \frac{n^3}{n^2} = n,npf(n)=n2n3=n, which grows unbounded as n→∞n →∞. Hence, the regularity condition is satisfied. **Step 4: Conclusion (Case 3 of Master Theorem)**

Since f(n)f(n)f(n) dominates npn^pnp (Case 3 of Master Theorem), the solution to the recurrence is determined by f(n)f(n)f(n):

T(n)=Θ(f(n))=Θ(n3)T(n) = \Theta(f(n)) = \Theta(n^3)T(n)=Θ(f(n))=Θ(n3)

Final result T(n)=Θ(n3)T(n) = \Theta(n^3)T(n)=Θ(n3)

Q2 T(n)=8T(2n)+c·n2,for n>1, and T(1)=1

Step 1: First Iteration

Expand T(n)T(n)T(n) by substituting T(n2)T\left(\frac{n}{2}\right)T(2n) using the same recurrence:

T(n)=8[T(n2)]+c·n2T(n) = 8\left[T\left(\frac{n}{2}\right)\right] + c \cdot n^2T(n)=8[T(2n)]+c·n2

Substituting T(n2)T\left(\frac{n}{2}\right)T(2n):

T(n2)=8T(n4)+c·(n2)2T\left(\frac{n}{2}\right) = 8T\left(\frac{n}{4}\right) + c \cdot \left(\frac{n}{2}\right)^2T(2n)=8T(4n)+c·(2n)2

Substitute this back:

T(n)=8[8T(n4)+c·(n2)2]+c·n2T(n) = 8\left[8T\left(\frac{n}{4}\right) + c \cdot \left(\frac{n}{2}\right)^2\right] + c \cdot n^2T(n)=8[8T(4n)+c·(2n)2]+c·n2

Simplify:

T(n)=82T(n4)+8·c·(n2)2+c·n2T(n) = 8^2 T\left(\frac{n}{4}\right) + 8 \cdot c \cdot \left(\frac{n}{2}\right)^2 + c \cdot n^2T(n)=82T(4n)+8·c·(2n)2+c·n2 T(n)=82T(n4)+2c·n2+c·n2T(n) = 8^2 T\left(\frac{n}{4}\right) + 2c \cdot n^2 + c \cdot n^2T(n)=82T(4n)+2c·n2+c·n2 T(n)=82T(n4)+3c·n2T(n) = 8^2 T\left(\frac{n}{4}\right) + 3c \cdot n^2T(n)=82T(4n)+3c·n2

Step 2: Second Iteration

Repeat the process by expanding T(n4)T\left(\frac{n}{4}\right)T(4n):

T(n4)=8T(n8)+c·(n4)2T\left(\frac{n}{4}\right) = 8T\left(\frac{n}{8}\right) + c \cdot \left(\frac{n}{4}\right)^2T(4n)=8T(8n)+c·(4n)2

Substitute into the previous equation:

T(n)=82[8T(n8)+c·(n4)2]+3c·n2T(n) = 8^2 \left[8T\left(\frac{n}{8}\right) + c \cdot \left(\frac{n}{4}\right)^2\right] + 3c \cdot n^2T(n)=82[8T(8n)+c·(4n)2]+3c·n2

Simplify:

T(n)=83T(n8)+82·c·(n4)2+3c·n2T(n) = 8^3 T\left(\frac{n}{8}\right) + 8^2 \cdot c \cdot \left(\frac{n}{4}\right)^2 + 3c \cdot n^2T(n)=83T(8n)+82·c·(4n)2+3c·n2

Step 3: General Pattern

After kkk iterations, the pattern emerges:

T(n)=8kT(n2k)+∑i=0k−18i⋅c⋅n2⋅(12i)2T(n) = 8^k T\left(\frac{n}{2^k}\right) + \sum_{i=0}^{k-1} 8^i \cdot c \cdot n^2 \cdot \left(\frac{1}{2^i}\right)^2T(n)=8kT(2kn)+i=0∑k−18i⋅c⋅n2⋅(2i1)2

Simplify the summation term:

∑i=0k−18i⋅c⋅n2⋅14i=c⋅n2∑i=0k−1(84)i=c⋅n2∑i=0k−12i\sum_{i=0}^{k-1} 8^i \cdot c \cdot n^2 \cdot \frac{1}{4^i} = c \cdot n^2 \sum_{i=0}^{k-1} \left(\frac{8}{4}\right)^i = c \cdot n^2 \sum_{i=0}^{k-1} 2^ii=0∑k−18i⋅c⋅n2⋅4i1=c⋅n2i=0∑k−1(48)i=c⋅n2i=0∑k−12i

The sum of a geometric series:

∑i=0k−12i=2k−1\sum_{i=0}^{k-1} 2^i = 2^k - 1i=0∑k−12i=2k−1

Thus:

T(n)=8kT(n2k)+c⋅n2⋅(2k−1)T(n) = 8^k T\left(\frac{n}{2^k}\right) + c \cdot n^2 \cdot (2^k - 1)T(n)=8kT(2kn)+c⋅n2⋅(2k−1)

Step 4: Termination of Iteration

When n2k=1\frac{n}{2^k} = 12kn=1, we have k=log2nk = \log_2 nk=log2n. Substitute kkk:

T(n)=8log2nT(1)+c⋅n2⋅(2log2n−1)T(n) = 8^{\log_2 n} T(1) + c \cdot n^2 \cdot (2^{\log_2 n} - 1)T(n)=8log2nT(1)+c⋅n2⋅(2log2n−1)

We know:

8log2n=(23)log2n=23⋅log2n=2log2n3=n38^{\log_2 n} = (2^3)^{\log_2 n} = 2^{3 \cdot \log_2 n} = 2^{\log_2 n^3} = n^38log2n=(23)log2n=23⋅log2n=2log2n3=n3

and:

2log2n=n2^{\log_2 n} = n2log2n=n

Substitute these:

T(n)=n3⋅T(1)+c⋅n2⋅(n−1)T(n) = n^3 \cdot T(1) + c \cdot n^2 \cdot (n - 1)T(n)=n3⋅T(1)+c⋅n2⋅(n−1)

Since T(1)=1T(1) = 1T(1)=1:

T(n)=n3+c⋅n3−c⋅n2T(n) = n^3 + c \cdot n^3 - c \cdot n^2T(n)=n3+c⋅n3−c⋅n2

Final Solution (Tight Bound):

The dominant term is n3n^3n3, so:

T(n)=Θ(n3)T(n) = \Theta(n^3)T(n)=Θ(n3)

Q3

The algorithm consists of two nested loops and an inner condition. We will analyze the number of steps executed step by step to compute the total.

```
Function ABC(A, n)
  for (i = 1 to n) do          → Outer loop
    for (j = n downto i + 1) do   → Inner loop
      if (A[j] < A[j - 1]) then  → Comparison
        SWAP(A[j], A[j - 1])   → Swap (conditional)
      end if
    end for
  end for
```

1. Outer Loop:

The outer loop runs from i=1i = 1i=1 to i=ni = ni=n.

Total iterations of the outer loop: n\text{Total iterations of the outer loop: }
nTotal iterations of the outer loop: n

2. Inner Loop:

The inner loop runs from j=nj = nj=n down to j=i+1j = i + 1j=i+1.

- For i=1i = 1i=1, the inner loop runs from j=nj = nj=n to j=2j = 2j=2, i.e., n−1n - 1n−1 iterations.
- For i=2i = 2i=2, the inner loop runs from j=nj = nj=n to j=3j = 3j=3, i.e., n−2n - 2n−2 iterations.
- For i=3i = 3i=3, the inner loop runs n−3n - 3n−3 iterations, and so on.
- For i=n−1i = n - 1i=n−1, the inner loop runs 111 iteration.
- For i=ni = ni=n, the inner loop runs 000 iterations.

The total number of iterations for the inner loop is:

∑i=1n−1(n−i)\sum_{i=1}^{n-1} (n - i)i=1∑n−1(n−i)

Simplify the summation:

∑i=1n−1(n−i)=n∑i=1n−11−∑i=1n−1i\sum_{i=1}^{n-1} (n - i) = n \sum_{i=1}^{n-1} 1 - \sum_{i=1}^{n-1} ii=1∑n−1(n−i)=ni=1∑n−11−i=1∑n−1i

- The first term:

∑i=1n−11=n−1\sum_{i=1}^{n-1} 1 = n - 1i=1∑n−11=n−1

- The second term:

∑i=1n−1i=(n−1)n2\sum_{i=1}^{n-1} i = \frac{(n-1)n}{2}i=1∑n−1i=2(n−1)n

Substitute these:

∑i=1n−1(n−i)=n(n−1)−(n−1)n2=n(n−1)2\sum_{i=1}^{n-1} (n - i) = n(n-1) - \frac{(n-1)n}{2} = \frac{n(n-1)}{2}i=1∑n−1(n−i)=n(n−1)−2(n−1)n=2n(n−1)

Thus, the inner loop executes n(n−1)2\frac{n(n-1)}{2}2n(n−1) iterations in total.

3. Steps per Inner Loop Iteration:

- **Comparison:** The condition A[j]<A[j−1]A[j] < A[j-1]A[j]<A[j−1] is evaluated once per iteration of the inner loop.
- **Swap:** A swap is performed only if the condition is true. In the worst case, every comparison results in a swap. Hence, each iteration involves one comparison and at most one swap.

Total Steps

1. **Number of Comparisons:** Each iteration of the inner loop performs one comparison:

   Total comparisons=n(n−1)2\text{Total comparisons} = \frac{n(n-1)}{2}Total comparisons=2n(n−1)

2. **Number of Swaps (Worst Case):** In the worst case, every comparison results in a swap:

   Total swaps (worst case)=n(n−1)2\text{Total swaps (worst case)} = \frac{n(n-1)}{2}Total swaps (worst case)=2n(n−1)

3. **Overall Steps:** Each iteration involves one comparison and, in the worst case, one swap. Therefore, the total steps in the worst case are:

Total steps (worst case)=2·n(n−1)2=n(n−1)\text{Total steps (worst case)} = 2 \cdot \frac{n(n-1)}{2} = n(n-1)Total steps (worst case)=2·2n(n−1)=n(n−1)

Final Answer:

The total number of steps executed in the worst case is:

n(n−1)n(n-1)n(n−1)

Q4 **Initialization:**

- Start by treating each character as a node with its frequency as the weight.

- **Sort Nodes by Frequency:**
  - Initially, the frequencies are: {B:5,C:15,D:20,A:25,E:35}\{B: 5, C: 15, D: 20, A: 25, E: 35\}{B:5,C:15,D:20,A:25,E:35}

- **Combine Two Lowest Frequencies:**
  - Combine B(5)B (5)B(5) and C(15)C (15)C(15) into a new node with a combined frequency of 5+15=205 + 15 = 205+15=20.
  - The new set of nodes is: {D:20,(BC):20,A:25,E:35}\{D: 20, (BC): 20, A: 25, E: 35\}{D:20,(BC):20,A:25,E:35}

- **Repeat Combination:**
  - Combine D(20)D (20)D(20) and BC(20)BC (20)BC(20) into a new node with a combined frequency of 20+20=4020 + 20 = 4020+20=40.
  - The new set of nodes is: {A:25,E:35,(BC−D):40}\{A: 25, E: 35, (BC-D): 40\}{A:25,E:35,(BC−D):40}

- **Combine Again:**
  - Combine A(25)A (25)A(25) and E(35)E (35)E(35) into a new node with a combined frequency of 25+35=6025 + 35 = 6025+35=60.
  - The new set of nodes is: {(BC−D):40,(AE):60}\{(BC-D): 40, (AE): 60\}{(BC−D):40,(AE):60}

Final compination Combine BC−D(40)BC-D (40)BC−D(40) and AE(60)AE (60)AE(60) into the root node with a combined frequency of 40+60=10040 + 60 = 10040+60=100

Huffman Tree Structure

The resulting tree can be visualized as follows:

scss
Copy code

```
              (100)
             /     \
          (40)      (60)
         /   \     /    \
     (BC-D)   D   A      E
    /     \
  (B)     (C)
```

Assign Binary Codes

- Start from the root and assign:
    - 000 to the left branch,
    - 111 to the right branch.

The resulting Huffman codes are:

Character	Binary Code
B	000
C	001
D	01
A	10
E	11

1. Q5 **Sort the Array**:
    - Sort the array SSS in ascending order.
    - Sorting takes O(nlog n)O(n \log n)O(nlogn).
    -
2. **Two-Pointer Technique**:
    - Use two pointers, left\text{left}left and right\text{right}right, initialized to the smallest (S[0]S[0]S[0]) and largest (S[n−1]S[n-1]S[n−1]) elements of SSS, respectively.
    - This step operates in O(n)O(n)O(n).

3. **Iterate Through the Array**:
    - At each step, compute the sum of the elements pointed to by left\text{left}left and right\text{right}right: sum=S[left]+S[right]\text{sum} = S[\text{left}] + S[\text{right}]sum=S[left]+S[right]
    - Calculate the absolute value: current_abs_sum=|sum|\text{current_abs_sum} = |\text{sum}|current_abs_sum=|sum|
    - Keep track of the smallest current_abs_sum\text{current_abs_sum}current_abs_sum and the corresponding pair (x,y)=(S[left],S[right])(x, y) = (S[\text{left}], S[\text{right}])(x,y)=(S[left],S[right]).
    -
4. **Update Pointers**:
    - If sum<0\text{sum} < 0sum<0: Move the left\text{left}left pointer one step to the right (left+=1\text{left} += 1left+=1) to increase the sum.
    - If sum>0\text{sum} > 0sum>0: Move the right\text{right}right pointer one step to the left (right−=1\text{right} -= 1right−=1) to decrease the sum.
    - Continue until left\text{left}left and right\text{right}right pointers meet.
5. **Output the Result**:
    - After completing the iteration, the pair (x,y)(x, y)(x,y) corresponding to the smallest |x+y||x + y||x+y| is the desired result.

الكود

```
#include <iostream>

#include <vector>

#include <algorithm>

#include <cmath>

using namespace std;


pair<int, int> findMinAbsSumPair(vector<int>& S) {

   sort(S.begin(), S.end());

   int left = 0, right = S.size() - 1;

   int minAbsSum = INT_MAX;

   pair<int, int> bestPair;
```

```
    while (left < right) {
        int currentSum = S[left] + S[right];
        int currentAbsSum = abs(currentSum);

        if (currentAbsSum < minAbsSum) {
            minAbsSum = currentAbsSum;
            bestPair = {S[left], S[right]};
        }

        if (currentSum < 0)
            left++;
        else if (currentSum > 0)
            right--;
        else
            break;
    }

    return bestPair;
}

int main() {
    vector<int> S = {-8, 4, 5, -10, 3};
    pair<int, int> result = findMinAbsSumPair(S);
    cout << "Pair: (" << result.first << ", " << result.second << ")\n";
    return 0;
}
```

Complexity Analysis

1. **Sorting**: Sorting the array takes O(nlog n)O(n \log n)O(nlogn).
2. **Two-Pointer Iteration**: The two-pointer approach traverses the array once, which takes O(n)O(n)O(n).

Overall Time Complexity:

O(nlog n)+O(n)=O(nlog n)O(n \log n) + O(n) = O(n \log n)O(nlogn)+O(n)=O(nlogn)

Q6 *Problem Analysis:*

We are given:

1. G=(V,E)G = (V, E)G=(V,E): A simple graph with nnn vertices.
2. Initially, all edges have a weight of 111.
3. Two edges are changed to have weights of 12\frac{1}{2}21.
4. The task is to determine the weight of the Minimum Spanning Tree (MST) of GGG.

Key Observations:

1. A simple graph with nnn vertices has a spanning tree of exactly n−1n-1n−1 edges.
2. For a graph where all edges have equal weight (111), any spanning tree of the graph will have the same weight: Weight of MST=(n−1)×1=n−1\text{Weight of MST} = (n-1) \times 1 = n-1Weight of MST=(n−1)×1=n−1
3. When two edges have their weights reduced to 12\frac{1}{2}21, these edges become more favorable for inclusion in the MST because MST algorithms prioritize edges with lower weights.

Approach to Solve:

1. Since the weights of two edges are reduced to 12\frac{1}{2}21, these edges will definitely be part of the MST (as they are cheaper than all other edges with weight 111).
2. The remaining n−3n-3n−3 edges in the MST will come from the other edges of weight 111.

Thus, the total weight of the MST becomes:

Weight of MST=2×12+(n−3)×1=1+(n−3)=n−2\text{Weight of MST} = 2 \times \frac{1}{2} + (n-3) \times 1 = 1 + (n-3) = n-2Weight of MST=2×21+(n−3)×1=1+(n−3)=n−2

Final Answer:

The weight of the MST is:

n−2\boxed{n-2}n−2