Mohab Yousef

COMP 3270

ID: 904154154

**Algorithm-1**

|  |  |  |
| --- | --- | --- |
| Step | Cost of each execution | Total # of times executed |
| 1 | 1 | 1 |
| 2 | 1 | n + 1 |
| 3 | 1 |  |
| 4 | 1 |  |
| 5 | 1 |  |
| 6 | 6 |  |
| 7 | 5 |  |
| 8 | 2 | 1 |

Multiply col.1 with col.2, add across rows and simplify

T1(n) = 1 + n+1 + + + + 6 + 5+ 2

= 1+n+1+ (n(n+1)/2) +1 + (n(n+ 1)/2) + ((n(n+ 1) (n+2))/6) +1 + (6(n)(n+ 1) (n+2)/6) + 5((n(n+1))/2) + 2

= (7/6) n^3 + 7n^2 + (41/6)n + 6

T1(n): O(n3)

**Algorithm-2**

|  |  |  |
| --- | --- | --- |
| Step | Cost of each execution | Total # of times executed |
| 1 | 1 | 1 |
| 2 | 1 | n+1 |
| 3 | 1 | n |
| 4 | 1 |  |
| 5 | 6 |  |
| 6 | 5 |  |
| 7 | 2 | 1 |

Multiply col.1 with col.2, add across rows and simplify

T2(n) = 1+n+1+n+ ((n(n+1))/2) +1+ (6n(n+1)/2) + (5n(n+1)/2) + 2

6n^2 + 8n + 5

T2(n): O(n2)

**Algorithm-3**

|  |  |  |
| --- | --- | --- |
| Step | Cost of each execution | Total # of times executed in any single recursive call |
| 1 | 4 | 1 |
| 2 | 11 | 1 |
| Steps executed when the input is a base case: 1. if L > U then  return 0 /\* zero- element vector \*/ 2. if L=U then  return max(0,X[L]) /\* one-element vector | | |
| First recurrence relation: T(n=1 or n=0) = when n= 1 is 4 and when n = 1 is 11 | | |
| 3 | 5 | 1 |
| 4 | 2 | 1 |
| 5 | 1 | n+1 |
| 6 | 6 | n |
| 7 | 7 | n |
| 8 | 2 | 1 |
| 9 | 1 | n+1 |
| 10 | 6 | n |
| 11 | 7 | n |
| 12 | 4 | 1 |
| 13 | 4 | logn(cost excluding the recursive call) |
| 14 | 5 | logn(cost excluding the recursive call) |
| 15 | 17 | 1 |
| Steps executed when input is NOT a base case: 13 | | |
| Second recurrence relation: T(n>1) = 2T(n/2) + 56 + 14n | | |
| Simplified second recurrence relation (ignore the constant term): T(n>1) = 2T(n/2) + 14n | | |

Solve the two recurrence relations using any method (recommended method is the Recursion Tree). Show your work below:

14n

/ \

14n/2 14n/2

/ \ / \

14n/4 14n/4 14n/4 14n/4

/ \ / \ / \ / \

log2n

T3(n) = 14n log2n + 14n

T3(n) = O(n log(n)

**Algorithm-4**

|  |  |  |
| --- | --- | --- |
| Step | Cost of each execution | Total # of times executed |
| 1 | 1 | 1 |
| 2 | 1 | 1 |
| 3 | 1 | n+1 |
| 4 | 7 | n |
| 5 | 7 | n |
| 6 | 2 | 1 |

Multiply col.1 with col.2, add across rows and simplify

T4(n) = 1 + 1 + n+1 + 7n + 7n + 2

= 15n + 5

T4(n): O(n)

This graph shows us that the Actual time taken follows and match the predicted time curve In algorithm 1, Algorithm 2, Algorithm 3, but not in Algorithm 4, since the actual time in algorithm 4 did not follow the predicted time curve for small values.