

University Computer Science

Name: _____ Score: _____

1. For $T(n) = 2T(n/2) + n$, one tree level costs –

- ☐ 1
- ☐ $n \log n$
- ☐ n
- ☐ $n/2$

2. Binary search recurrence is –

- ☐ $T(n/2) + 1$
- ☐ $2T(n/2) + n$
- ☐ $2T(n/2) + 1$
- ☐ $T(n-1) + n$

3. The exchange argument shows –

- ☐ greedy always fails
- ☐ nothing
- ☐ optimal can afford greedy choices
- ☐ greedy is fast

4. The DP design is –

- ☐ the cache library
- ☐ the loop structure
- ☐ recurrence + base cases + fill order
- ☐ the recurrence only

5. The light edge crossing a cut is –

- ☐ always the heaviest
- ☐ irrelevant
- ☐ never in any MST
- ☐ in some MST – safe to accept

6. A reduction A to B gives –

- ☐ an approximation
- ☐ a B-solver from an A-solver

- ☐ nothing
 - ☐ an A-solver from a B-solver
7. $T(n) = 4T(n/2) + n$ solves to –
- ☐ $\log n$
 - ☐ n^2
 - ☐ $n \log n$
 - ☐ n
8. Sorted input with first-element pivots costs quicksort –
- ☐ $\Theta(n^2)$
 - ☐ $\Theta(n \log n)$
 - ☐ $\Theta(n)$
 - ☐ $\Theta(\log n)$
9. Earliest-finish on the course activities picks –
- ☐ 4 activities
 - ☐ 3 activities
 - ☐ 2 activities
 - ☐ 5 activities
10. Naive $\text{fib}(30)$ makes – calls
- ☐ 2^{30}
 - ☐ 29
 - ☐ 2,692,537
 - ☐ 1,024

University Computer Science

1. **n**

Level i has 2^i nodes of $n/2^i$: the product is n at every level.

2. **$T(n/2) + 1$**

One recursive call, constant combine: $\Theta(\log n)$.

3. **optimal can afford greedy choices**

Swap optimal picks for greedy ones losslessly: same size, still feasible.

4. **recurrence + base cases + fill order**

Three parts: what a cell means, where it starts, which direction to fill.

5. **in some MST – safe to accept**

Tree + light edge - cycle other edge keeps or lowers the weight.

6. **an A-solver from a B-solver**

The translator maps A-instances to B-instances: solving B solves A.

7. **n^2**

Leaves dominate: $n^{\log_2 4} = n^2$.

8. **$\Theta(n^2)$**

$T(n) = T(n-1) + n$: each partition peels one element.

9. **4 activities**

(1,4), (5,7), (8,11), (12,16) - four, provably optimal.

10. **2,692,537**

The call count is itself Fibonacci-exponential; memoised it is 29 subproblems.