Week 9 Mon.

1. Which of the following statements are correct?

The best-case running time of A is Ω(g(n)) implies the running time of A is Ω(g(n)).✅

The best-case running time of A is O(g(n)) implies the running time of A is O(g(n)).

The worst-case running time of A is Ω(g(n)) implies the running time of A is Ω(g(n)).

The worst-case running time of A is O(g(n)) implies the running time of A is O(g(n)).✅

Week 9 Thu.

1. The loop invariant of Moore and Boyer's Algorithm for the Majority Element Problem is:

Remove *count* number of *candidate* from A[1 .. i-1], the remaining array does not contain any majority element.

Before each iteration, let A' be the remaining array after removing *count* number of *candidate* from A[1 .. i-1] . We can discuss the maintenance of loop invariant in the following three cases. Please fill in A or B in each blank to complete the argument.

1. Before the iteration, *count* == 0. Then after the iteration: __**A**__

2. Before the iteration, *count*≠0 and *candidate* == A[i]. Then after the iteration: __**A**__

3. Before the iteration, *count*≠0 and *candidate*≠A[i]. Then after the iteration: __**B**__

A. *count* is increased by 1. A' remains the same, containing no majority element.

B. *count* is decreased by 1. One *candidate* and one A[i] are added into A'.

The *candidate*, A[i], or any other element in A' cannot be the majority element.

Week 10 Mon.

1. Let $A[1..n]$ be an array of n distinct numbers.

If $i < j$ and $A[i] > A[j]$, then the pair (i, j) is called an inversion of A.

Use global variable *count* to denote the number of inversions. How to modify function MERGE to compute *count*?

```
MERGE(A, p, q, r)
n1 = q - p + 1
n2 = r - q
//create arrays L[1..n1 + 1] and R[1..n2 + 1]
for i = 1 to n1
      L[i] = A[p + i - 1]
for j = 1 to n2
      R[j] = A[q + j]
L[n1 + 1] = ∞
R[n2 + 1] = ∞
i = 1
j = 1
for k = p to r
    if L[i] ≤ R[j]
          A[k] = L[i]
          i = i + 1
    else A[k] = R[j]
          j = j + 1
```

Week 10 Thu.

1. D&C algorithms can only be implemented recursively.

- True
- False ✅

2. The efficiency of D&C algorithms depends only on the number of subproblems and time for the divide and combine steps

- True
- False✅

Week 11 Mon.

If a divide-and-conquer algorithm A runs in $T(n) = aT\left(\frac{n}{b}\right) + f(n)$.

1. $T(n) = \Omega(n^{\log_b a})$

- True ✅
- False

2. If a>1, the running time of A can be bounded by a logarithmic function (i.e., $T(n) = O(lg\ n)$).

- True
- False ✅

3. When a < b and $f(n) = \Theta(n)$, then $T(n) = \Theta(n)$.

- True ✅
- False

Week 11 Thu.

Searching for a value x in an unsorted array A consisting of n elements. Strategy: pick a random index i into A. If $A[i] = x$, then we terminate; otherwise we continue the search by picking another index, until we find an index j such that $A[j] = x$ or we have checked all elements in A.

1. Suppose there is exactly one index i such that $A[i] = x$. What is the expected number of indices into A we must pick before we find x?

- n^2
- n ✅

2. Suppose there are no indices i such that $A[i] = x$. What is the expected number of indices into A that we must pick before we have checked all elements of A?

- $n(\ln n + O(1))$ ✅
- n^2

Week 12 Mon.

In each of the following questions, you are given a rank array of 10 applicants. Find the number of hired applicants.

Note: a bigger rank denotes a better applicant.

1. 4 2 1 5 3 6 8 7 9 10

- 4
- 5
- 6 ✅
- 7

2. 2 4 7 5 7 7 3 8 1 6

- 3
- 4 ✅
- 5
- 6

Week 12 Thu.

1. The expected number of hired applicants of HIRE-ASSISTANT is $O(\ln n)$ for any input distribution.

- True
- False ✅

2. The expected number of hired applicants of RANDOMIZED-HIRE-ASSISTANT is $O(\ln n)$ for any input distribution.

- True ✅
- False

Week 13 Mon.

What is the recurrence for the D&C algorithm of the Maximum Subarray Problem?

- $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(1)$
- $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$ ✅
- $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n^2)$

What is the running time of this algorithm?

- $\Theta(\lg n)$
- $\Theta(n)$
- $\Theta(\mathrm{n} \lg n)$ ✅
- $\Theta(n^2)$

Week 13 Thu.

1. Consider the DP algorithm for the parenthesization problem. Denote the number of alternative ways of parenthesization for a sequence of n matrices by P(n).

$$P(n) = \begin{cases} 1 & n = 1 \\ \sum_{k=1}^{n-1} P(k)P(n-k) & n > 1 \end{cases}$$

P(5) = **14**

Week 14 Mon.

1. Both D&C algorithms and DP algorithms view a problem as a collection of subproblems.

- True ✅
- False

2. We normally determine the running time of both D&C algorithms and DP algorithms by solving a recurrence.

- True
- False ✅

Week 14 Thu.

Give a greedy-choice candidate for the activity selection problem.