CHAPTER 3 – PROBLEMS

Problem 1. Given an array A as follow:

$$A =< -9, -9, -5, -2, 0, 3, 7, 7, 10, 15 >$$

a. Calculate the number of comparison operations to find $x = -9$ using two algorithms: sequential search and binary search (only count the comparison between the search key and array's elements). Give your comments about the results.
Sequential search: 1
Binary search: 2
Comment: Sequential search is better than binary search in this case because the search key is the first element of array.

b. In case of using binary search, which value is returned (0 or 1)?
→ 1

Problem 2. During a binary search, which elements in the array $A =< 4, 8, 12, 14, 20, 24 >$ are compared to the *key* when the *key* is:

a. 2 → 12, 4
b. 8 → 12, 4, 8
c. 15 → 12, 20, 14

Problem 3. Given an array A of n integers. Write a search algorithm as follow:

- *Step 1:* Sort A using any sort algorithm that you like
- *Step 2:* Search for an integer x in A using sequential search, return i if $A[i] == x$ and returns -1 if $A[i] > x$.

```
int Search(int a[], int n, int key){
      // Step 1
      QuickSort(a, n);
      // Step 2
      int i = 0;
      while (i < n && a[i] < key) i++;
      if (i < n && a[i] == key) return i;
      else return -1;
}
```

a. Is this algorithm better the original sequential search algorithm which do not sort the array A beforehand? (Hint: observe the case that there exists x in A and the case that x does not appear in A)
No. Because Quicksort algorithm already takes O(nlogn) running time while the original sequential search only takes O(n).

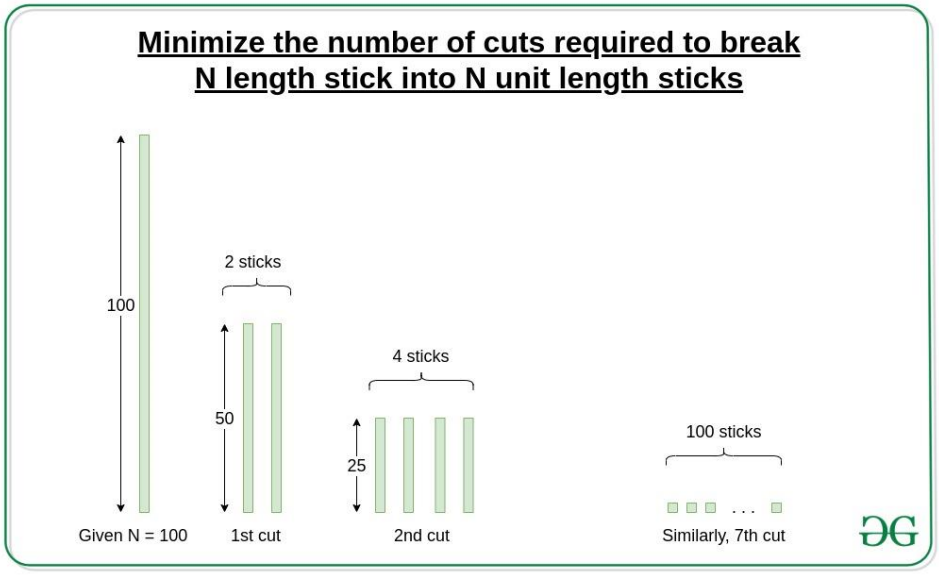
b. Is this algorithm faster than a binary search algorithm? → No

Problem 4. ***Cutting sticks.*** A stick n meters long needs to be cut into n 1-m pieces. Outline an algorithm that performs this task with the minimum number of cuts if several

pieces of the stick can be cut at the same time. Also give a formula for the minimum number of cuts.

Each time, we try to cut the stick into 2 possible longest pieces:
1st cut: N = 2 pieces x N/2 m
2nd cut: N = 2 pieces x (2 pieces x N/4 m)
3rd cut: N = 2 pieces x (2 pieces x (2 x N/8 m))
…
k^{th} cut: N = 2 pieces x (2 pieces x (2 x … x (2 x $N/2^k$ m)))
where $N/2^k = 1 \rightarrow k = \lceil \log_2 N \rceil$

For example, when N = 100:



Problem 5. The time efficiency of sequential search does not depend on whether a list is implemented as an array or as a linked list. Is it also true for searching a sorted list by binary search?
It is not true. On a linked list, finding the middle element takes O(n) in every time, whereas it is only O(1) on a sorted array.

Problem 6. ***Picture guessing***. A version of the popular problem-solving task involves presenting people with an array of 42 pictures—seven rows of six pictures each— and asking them to identify the target picture by asking questions that can be answered yes or no. Further, people are then required to identify the picture with as few questions as possible. Suggest the most efficient algorithm for this problem and indicate the largest number of questions that may be necessary.

It can be solved using a binary search-based algorithm:
- Divide the original array (42 pictures) into 2 halves (21 pictures) by asking "Is the target picture in the top half of the array?"
- Continue the dividing strategy to narrow down the number of pictures to search (21→11→5→2→1) until you get the answer.

➔ Max questions = $\log_2(42) \approx 5.39$

Problem 7. An array $A[0..n-2]$ contains $n-1$ integers from 1 to n in increasing order. (Thus one integer in this range is missing.) Design the most efficient algorithm you can to find the missing integer and indicate its time efficiency.

1. Firstly, calculate the sum $S1 = 1 + 2 + \cdots + n = \frac{n(n+1)}{2}$. This is the expected sum if no integer is missing. → O(1)
2. Calculate the actual sum S2 of array A by using a loop iterating through A and calculate sum of all elements in A.

$$S2 = A[0] + A[1] + \cdots + A[n-2]$$

→ O(n)
3. The missing integer is: $x = S1 - S2$

The basic operation is addition inside the loop of step 2

➔ O(n)