The University of Newcastle
School of Electrical Engineering and Computer Science

COMP2230 Algorithms

Tutorial Week 5 Solutions

16th – 20th August 2021

Tutorial

The coin-weighing problem:
There are n coins that look identical, but one of them is slightly heavier or slightly lighter than the other coins. The task is to identify the bad coin and to determine whether the coin is heavier or lighter using only a pan balance.

1. Sequential search can be used with the same efficiency in the case the list of elements is implemented as an array or as a linked list. Is this also true for binary search?

Solution: No. If the list is implemented as an array, the worst-case time complexity of binary search is $C_{bin}(n) = \Theta(lg\ n)$. In a linked list of size n, it would take $n/2$ steps to reach the middle element, starting from the first element and following the pointers (assuming that the list is sorted). Thus if the list is implemented as a linked list, the worst-case time complexity of binary search is $C'_{bin}(n) = n/2 + n/4 + n/8 + \ldots = \Theta(n)$.

2. Write an algorithm to solve the coin-weighing problem for n = 5 using three weighings in the worst case.

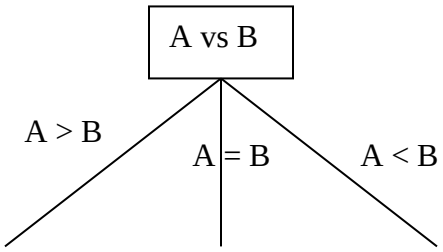
Solution: We present an algorithm that uses 3 weighings in the worst case and 2.8 weighings on average. Can you construct an algorithm with a better average-case complexity?

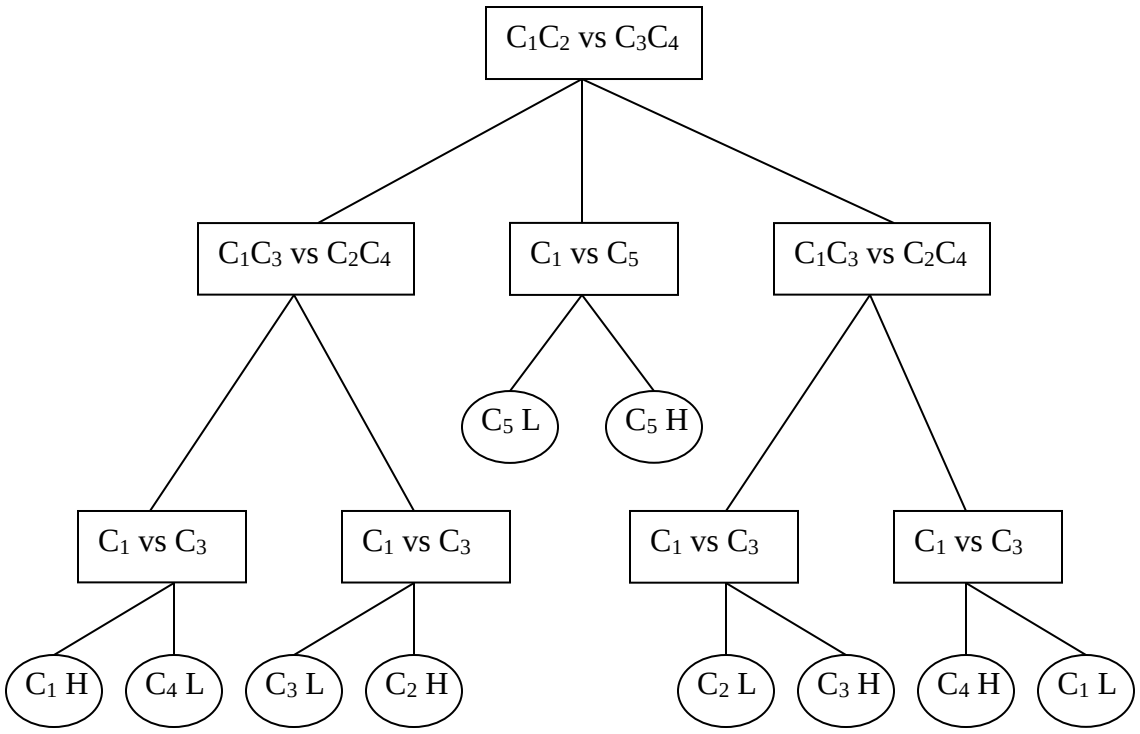
Algorithm:

```
// weigh C1 and C2 against C3 and C4
if (C1 + C2) > (C3 + C4)
    // weigh C1 and C3 against C2 and C4
    if (C1 + C3) > (C2 + C4)
            // weigh C1 against C3
            if C1 > C3
                    C1 is the heavy coin
            else
                    C4 is the light coin
    else
```

```
            // weigh C1 against C3
            if C1 > C3
                  C3 is the light coin
            else
                  C2 is the heavy coin
else if (C1 + C2) = (C3 + C4)
    // weigh C1 against C5
    if C1 > C5
                  C5 is the light coin
            else
                  C5 is the heavy coin
else
    // weigh C1 and C3 against C2 and C4
    if (C1 + C3) > (C2 + C4)
            // weigh C1 against C3
            if C1 < C3
                  C3 is the heavy coin
            else
                  C2 is the light coin
    else
            // weigh C1 against C3
            if C1 < C3
                  C1 is the light coin
            else
                  C4 is the heavy coin
```

A decision tree for this algorithm is given below. The following convention is used: the right branch corresponds to the right-hand side heavier, the middle branch corresponds to balanced sides, and the right branch corresponds to the right-hand side heavier (see the picture below),



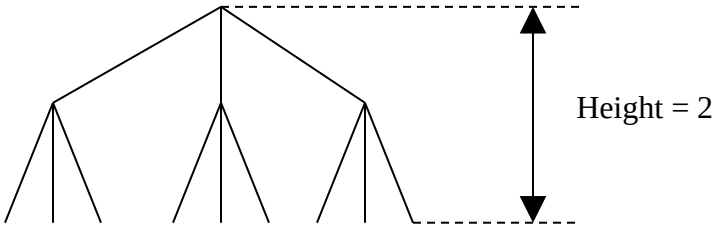


Average number of weighings for the above algorithm is 2.8.

3. Prove that the coin weighing problem for $n = 5$ cannot be solved using less than 3 weighings in the worst case.

Solution: Assume that there is an algorithm to determine the bad coin and whether it is heavier or lighter with only two weighings. We can describe the algorithm by using a decision tree, where each node represents a weighing, and each node has at most three children, one for each possible outcomes (left side heavier, balanced, or right side heavier). Since there are only two weighings allowed, the decision tree has height at most 2, and thus can have at most 9 leaves. However, the coin-weighing problem for n = 5 has 10 possible outcomes (1H, 1L, 2H, 2L, 3H, 3L, 4H, 4L, 5H, 5L). We have a contradiction and we can conclude that every algorithm for 5-coin-

weighing problem requires at least 3 weighings.

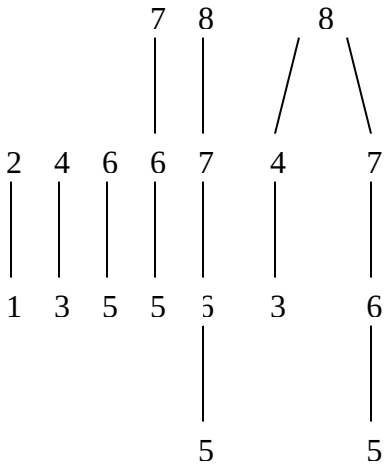


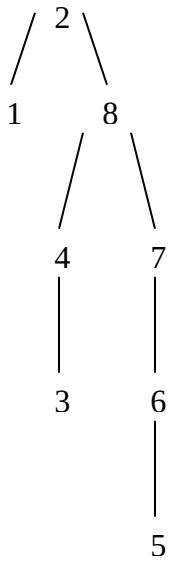
Decision tree for 5-coin-weighing algorithm

4. Show the trees that result after the following statements are executed

```
For i=1 to 8
     Makeset1(i)

union1(1,2)
union1(3,4)
union1(5,6)
union1(5,7)
union1(5,8)
union1(4,8)
union1(3,2)
```

```
Makeset1(i){
     parent[i]=i
}

findset1(i){
     while(i!=parent[i])
          i=parent[i]
     return i
}

mergetrees1(i,j){
     parent[i]=j
}

union1(i,j){
     mergetrees1(findset1(i),findset(j))}
```





Workshop

5. What happens if union1 is erroneously called with i and j in the same tree?

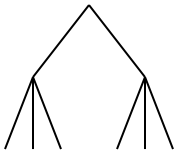
Solution: Text pg674 "*The trees are unchanged. Sind findset1[i] will be equal to findset1[j], mergetrees1 will be called with both arguments equal to the root of the tree. Thus mergetrees1 will set the parent of the root equal to the root. But this was already the value of the parent of the root.* ***"***

6. Write an algorithm to solve the coin-weighing problem for n = 4 using three weighings in the worst case.

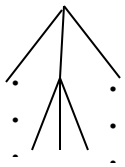
Solution idea: Weigh C_1 against C_2. If one is lighter, weigh C_1 against C_3; if they balance, weigh C_1 against C_3 again; etc. What is the average number of weighings?

7. Prove that the coin weighing problem for n = 4 cannot be solved using less than 3 weighings in the worst case.

Solution: In the text, page 676 (exercise 24, Section 4.1)
"Suppose that there is an algorithm that can solve the four-coin problem in two weighings in the worst case. Then the decision tree for the algorithm has height 2. The first weighing either compares two coins against two coins or one coin against one coin. The following figure shows that if the algorithms begins by weighing two coins against 2 coins, the decision tree can account for at most six outcomes; however, eight outcomes are possible:



An edge from a node to a left child means that the left pan weighs less than the right pan. An edge from a node to a right child means that the left pan weighs more than the right pan. A middle edge means that the pans balance. Notice that when the algorithm weighs two coins against two coins, they cannot balance. Therefore, no algorithm can begin by weighing two coins against two.
Similarly, the following figure shows that if the algorithms begins by weighing one coins against one coins, the decision tree can account for at most three outcomes; however, four outcomes are possible:



Therefore, no algorithm can begin by weighing one coin against one coin, and there is no algorithm that can solve the four-coin problem in two weighing in the worst case."

8. What is wrong with the following argument?
To solve the coin-weighing problem for n=12, we need at least 4 weighings if we start by weighing 4 coins against 4 coins. If we weigh 4 coins against 4 coins and they balance, we must then determine the bad coin from the remaining 4 coins. But we know from the previous exercise that determining the bad coin from among 4 coins requires at least 3 weighings in the worst case. Thus we require at least 4 weighings.

Solution: If we weigh 4 coins against 4 coins and they balance, the problem does not reduce to the problem of finding a bad coin from among 4 coins, as we have 8 good coins that we can use in our weighings and solve the latter problem in at most 2 weighings.

9. You are given 42 pictures arranged in 7 rows and 6 columns. Your task is to construct a serious of yes/no questions so that by getting correct answers to your questions you can identify the picture with as few questions as possible.

More exercises

10. Write an algorithm to solve the coin-weighing problem for n = 12 using three weighings in the worst case.

Solution idea: First prove that you have to start with 4 coins against 4 coins; if you do anything else, you will not have enough leaves in one of the 3 subtrees to cover all the possible outcomes.

11. Prove that the coin weighing problem for n = 12 cannot be solved using less than 3 weighings in the worst case.

Solution: Follows directly from the solution to question 3.

12. Write an optimal algorithm to solve the coin-weighing problem for n = 13. Prove that your algorithm is optimal.

13. Show that if m<n, after executing Algorithms makeset1, findeset1 and union1 the maximum height of a tree is m.

Solution: text pg 674 ***"****When union executes, the height of the tree that results can be at most 1 more than the maximum height of the two trees that were merged. Since union is called at most m times and all tree began with height 0, the maximum height of a tree is m****"***