Selection Problems

```
int FindMax(int[] list,int low, int high){
    int max = low;
    for(int i=low+1;i<=high; i++)
       if (list[i]>list[max])
              max = i;
    return max;
 }
```

What is the cost? Is this optimal?

Proof of Lower Bound

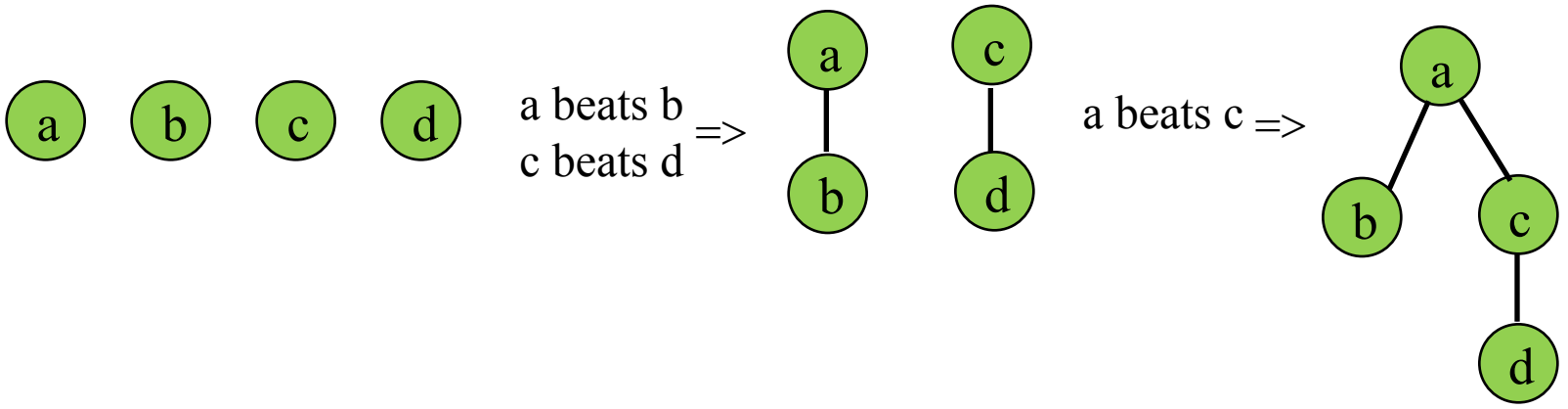
```
int FindMax(int[] list,int low, int high){
    int max = low;
    for(int i=low+1;i<=high; i++)
       if (list[i]>list[max])
             max = i;
    return max;
 }
```

The winner must compare against all other elements, so there must be $n - 1$ comparisons.

What is the flaw of this argument?

Partial Ordered Sets (Posets)

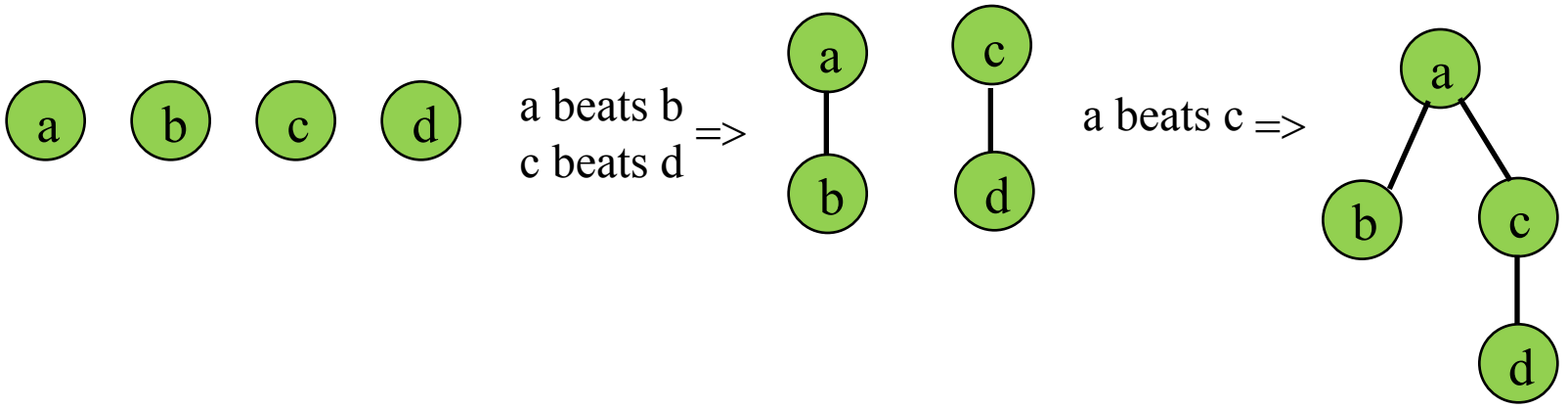
Suppose n opponents have matches to determine the winner.



What posets with 4 elements find a winner besides this one? (ignore symmetries)

Lower Bound of FindMax

To find the max, we must build a poset having one max and n − 1 losers, starting from a poset of n singletons. We wish to connect the elements of the poset with the minimum number of links.



Why can we now argue that this will require n-1 comparisons?

Selection Problems – Finding 2nd Best

Algorithm: Find the max and discard it and then find the max of what is left

What is the cost? Is this optimal?

Proof of lower bound– Finding 2nd Best

Lower bound:

Anyone who lost to anyone who is not the max cannot be second.

So, the only candidates are those who lost to max.

Findmax might compare max to $n - 1$ others.

Thus, we might need $n - 2$ additional comparisons to find second

What is wrong with this argument?

Selection Problems – Finding 2nd Best

Alternative: Divide and conquer

Break the list into two halves.

Use Findmax on each half.

Compare the winners.

Use Findmax on the winner's half for second.

Compare that second to second winner.

What is the cost? Is this optimal?

Selection Problems – Finding 2nd Best

Alternative: Divide and conquer

Break the list into two halves.

Use Findmax on each half.

Compare the winners.

Use Findmax on the winner's half for second.

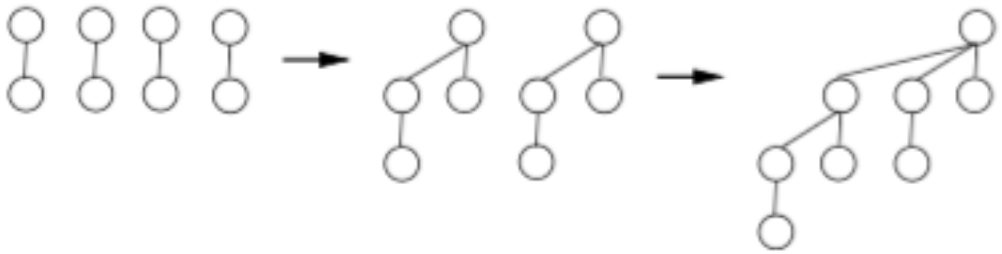
Compare that second to second winner.

What if we break the list into four pieces? Eight pieces?

Binomial Trees

The only candidates for second are losers to the eventual winner. Have the contest form a binomial tree:
A binomial tree of height m has 2m nodes organized as:

- a single node, if m = 0, or
- two height m − 1 binomial trees with one tree's root becoming a child of the other.

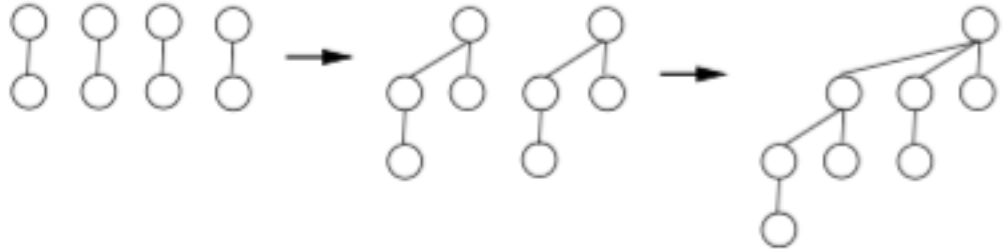


If n nodes, how many candidates for second best?

Binomial Trees

Algorithm:
Build the binomial tree.

Compare the [logn] children of the root for second.



Cost?

Adversarial Lower Bounds Proof

The algorithm asks the adversary for information about the input. The adversary may never lie.

Imagine that the adversary keeps a list of all possible inputs. When the algorithm asks a question, the adversary answers, and crosses out all remaining inputs inconsistent with that answer.

The adversary is permitted to give any answer that is consistent with at least one remaining input.

Explain how an adversary would make an algorithm work as hard as possible in a game of Hangman?

Adversary for 2^{nd} Max

At least $n - 1$ values must lose at least once so at least $n - 1$ compares.

Have k direct losers to the winner which must be compared for possibilities for 2^{nd} max.

There must be at least $n + k - 2$ comparisons.

What question are we asking for a lower bound proof ?

Adversary for 2nd Max

Call the strength of element L[i] the number of elements L[i] is (known to be) bigger than. If L[i] has strength a, and L[j] has strength b, then the winner has strength a + b + 1.

The adversary wants to minimize the rate at which an element gets stronger.

So on a contest between two candidates which one will the adversary pick as the winner? Why?

Adversary for 2^{nd} Max

So at each contest, an element's strength at most doubles. After k comparisons an elements strength is less than or equal to 2^k.

What is the lowest value we can have for k as losers to the max?