§7 Lower Bounds

§7.1 Adversarial Arguments

We can find the maximum element in an unsorted array $A[1..n]$ in $O(n)$ time.

How can we argue that this is optimal?

Idea: Show that any algorithm which executes a number of steps that is "too small" can be fooled by at least one input.

We show that with only $n-1$ steps, we cannot find the maximum element in an array $A[1..n]$.

Reading the number at position i in A corresponds to one step. If we can execute only $n-1$ steps, there is a position $1 \le k \le n$ that cannot be read. Therefore, the algorithm cannot know the value $A[k]$. Thus, the algorithm cannot find the maximum value in A.

The question we ask is: can I get another value from A (please)?

206B

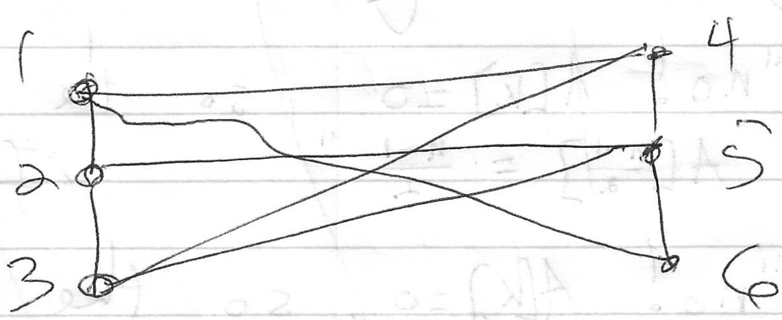
Indeed, an adversary could return the value $\underline{2}$ for all positions in A (except k).

If the algorithm says "the max is $A[k]$", the adversary says "no! $A[k]=1$."

If the algorithm says "the max is 2", the adversary says "no! $A[k]=3$".

Here, One operation corresponds to read ~~the~~ a number in A.

the question we ask is can I get another value from A (please)?



Recall: Let

$$f : \mathbb{N} \longrightarrow \mathbb{R}^+$$

$$g : \mathbb{N} \longrightarrow \mathbb{R}^+$$

be two functions. We say that f is Ω of g if there exist a constant $c \in \mathbb{R}^+$ and a number $K \in \mathbb{N}$ such that $f(n) \geq c \cdot g(n)$ for all $n \geq K$. We write $f(n) = \Omega(g(n))$.

So we need $\Omega(n-1) = \Omega(n)$ steps to find the maximum element in an array. Hence, scanning the array to find the maximum element in $O(n)$ time is optimal.

Another example:

Let $\mathbb{A}$ be an algorithm which finds the median in an array $A[1..n]$. Explain why $\mathbb{A}$ cannot take less than n steps.

208 B

Indeed, assume n is odd, for each i (except k), an adversary could return $A[i] = i$.

If the algorithm says "the median is $A[k]$", the adversary says

- "no! $A[k]=0$, so the median is $A[\frac{n-1}{2}] = \frac{n-1}{2}$" if $k \geq \frac{n+1}{2}$
- "no! $A[k]=0$, so the median is $A[\frac{n+1}{2}] = \frac{n+1}{2}$" if $k < \frac{n+1}{2}$

If the algorithm says "the median is $A[j] = j$" for $j \neq k$, the adversary says

- "no! $A[k]=k$, so the median is $A[\frac{n+1}{2}] = \frac{n+1}{2}$" if $j \neq \frac{n+1}{2}$
- "no! $A[k]=n+1$, so the median is $A[\frac{n+3}{2}] = \frac{n+3}{2}$" if $j = \frac{n+1}{2}$ and $k < \frac{n+1}{2}$
- "no! $A[k]=0$, so the median is $A[\frac{n-1}{2}] = \frac{n-1}{2}$" if $j = \frac{n+1}{2}$ and $k > \frac{n+1}{2}$

In $n-1$ steps only, there is a position $1 \le k \le n$ that cannot be read. Therefore, A cannot find the median (similar reasonning).

So we need $\Omega(n-1) = \Omega(n)$ steps to solve this problème. So the Selection algorithm we have seen in class is optimal.

The question we ask is: can I get another value from A ?

<u>One more example:</u>

Test the connectivity of an undirected graph $G = (V, E)$.

<u>Algo</u>:

```
for all u ∈ V:
        visited(u) = false
v = arbitrary vertex of G
explore(v)
for all u ∈ V:
        if visited(u) = false
                return false
return true
```

```
explore(v):
    visited(v) = true
    for each {u,v} ∈ E:
        if visited(u) = false
            explore(u)
```

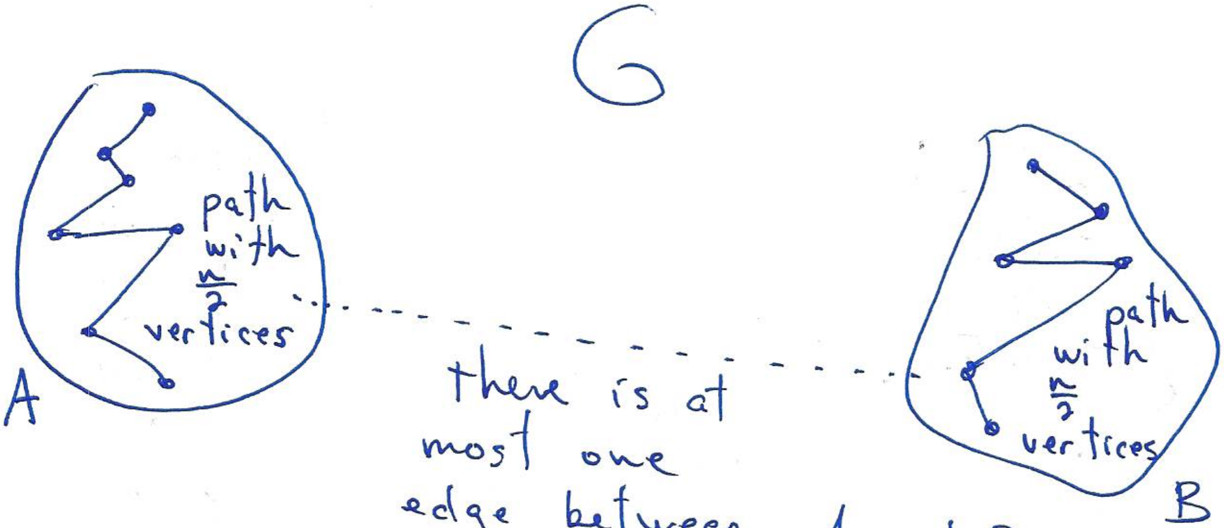
With the adjacency list representation, this algorithm takes $O(|V|+|E|)$ time.

With the adjacency matrix representation, this algorithm takes $O(|V|^2)$ time. Can we do better?

We will show that we need $\Omega(|V|^2)$ steps to solve this problem with the adjacency matrix representation.

The question we ask is: is there an edge between vertices u and v?

We show that $\frac{n}{2} \times \frac{n}{2} - 1$ steps are not sufficient to test the connectivity of an undirected graph.



there is at most one edge between A and B.

to Test whether or not G is connected, we need to ask if $\{u, v\}$ is an edge for all $u \in A$, $v \in B$. Otherwise, we might miss the "only" edge between A and B. So we need to test at least $\frac{n}{2} \times \frac{n}{2}$ edges. So it takes $\Omega\left(\frac{n^2}{4}\right) = \Omega(n^2)$ time.

What is the running time if we take the adjacency list representation?

In this case, the question we ask becomes: can I get another edge from vertex v?