Review:

$n! > 2^n$: Prove by induction for $n \geq 4$

Base Case : $4! = 24$ $\quad 2^4 = 16$.
$4! > 2^4$.

Assume $n! > 2^n$: we will show $(n+1)! > 2^{n+1}$

$(n+1)! = (n+1) \cdot n!$
$> (n+1) \cdot 2^n$ (by assumption)
$> 2 \cdot 2^n$ (since $n \geq 4$).
$= 2^{n+1}$

Maximal Interval Selection

Input : Collection of intervals.

Output : Subcollection of maximal size of non-overlaping intervals.

An Algorithm :
While an interval remain :
select the earliest ending
Delete overlap.

From HW, some algorithm that are not feasible.

① Choose the interval that ends last.

② Choose the interval that starts first

③ Choose the shortest

④ While overlap delete the biggest course

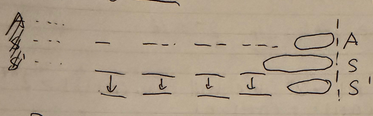
⑤ While overlap delete the course that overlap with it

---

⑥ Selecting the ~~lastest~~ latest start works.

Exchange Argument

we agree.



- Doesn't get worse : 1-1 swap.
- Feasibility : Nothing to the left because we selected the latest starting, and they agree with our selections before so nothing to the right
- No new earlier disagreement created.

For i in range (n) :
for j in range (n) :
do $i \times j$ worse.

$$\sum_{i=1}^{n}\sum_{j=1}^{n} 1 \leq \sum_{i=1}^{n}\sum_{j=1}^{n} i \times j \leq \sum_{i=1}^{n}\sum_{j=1}^{n} n \times n = n^4.$$

$$\Theta(n^4) = \sum_{i=1}^{n}\sum_{j=1}^{n} i \times j \geq \sum_{i=\frac{n}{2}}^{n}\sum_{j=\frac{n}{2}}^{n} i \times j \geq \sum_{j=\frac{n}{2}}^{n}\sum_{j=\frac{n}{2}}^{n} \frac{n}{2} \times \frac{n}{2} = \frac{n^4}{16}.$$

Merge (L, R)

out = [ ]
L.idx = 0.
R.idx = 0.
while (len(out) < len(L) + len(R)) : = for i in range (len(L)+len(R)).
if (L.idx < len(L) && (R.idx = len(R) || L[L.idx] ≤ R[R.idx])
add L[L.idx] to out.
L.idx ++
else
add R[R.idx] to out
R.idx ++.
return out.

$\Theta(1)$.

Only depends on $\|L\| + \|R\|$, nothing to do with context.
→ runtime = $n+m$.

Bubble Sort (A) :

While the 1st isn't sorted : // until I do an iter ~~that~~ without change anything
for i in range (n-1) :
if A[i+1] < A[i] :
swap them.

- ~~takes~~ When the list is sorted, it takes linear time.
  ⇒ Best Case upper bound : $O(n)$.
  How do we know no input gives a faster time ? while loop goes through each element to check.
  ⇒ Best Case lower bound : $\Omega(n)$
- When the list is inverse, it takes n^2 time.
  ⇒ Worst Case lower bound = $\Omega(n^2)$
  How do we know no input gives a slower time ? Prove by induction...?
  ⇒ Worst Case upper bound = $O(n^2)$.