CS124

Lecture 3

ex

n^n is $O(2^n)$ NO

$\lim \frac{n^n}{2^n} \to \infty$

ex

My alg is $O(n^2)$

Your alg is $O(n^2 \log^2 n)$

So mine is faster NO

→ const

→ not tight bound

$O()$ ≤ asymptotically (up to constant factors)

$f(n)$ is $o(g(n))$ ≪ asymptotically

if $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$

difference must increase as $n \to \infty$

cannot be a constant difference

Ω ≥ asymptotically

$f(n)$ is $\Omega(g(n))$

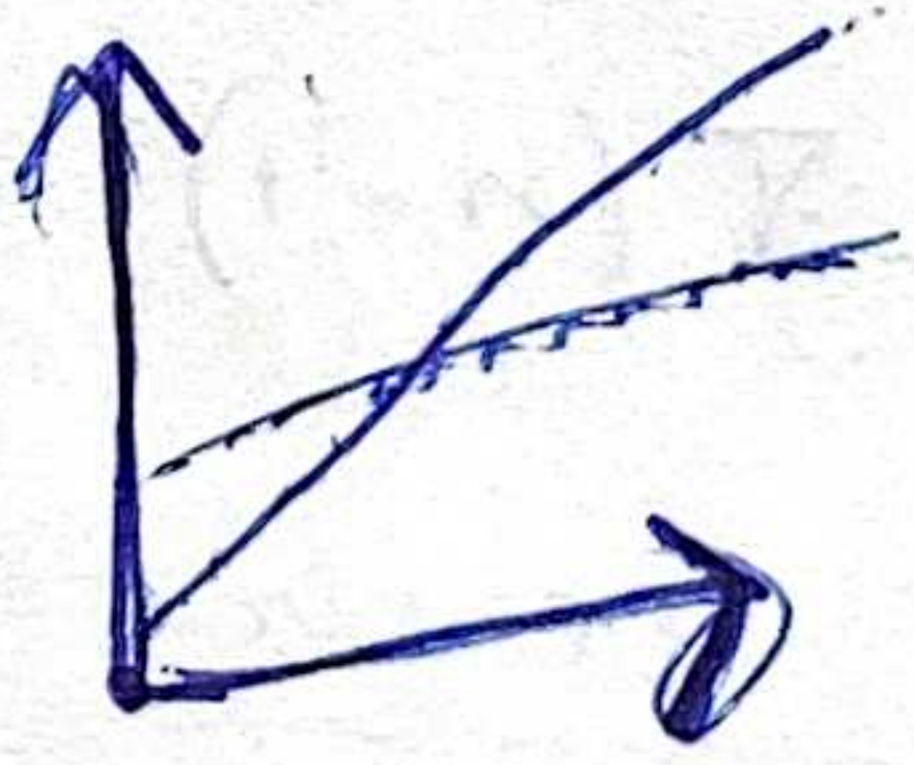
if $\exists\, c > 0,\ N > 0$, s.t [redacted]

$f(n) \geq c\, g(n) \quad \forall n \geq N$

≫ asymptotically

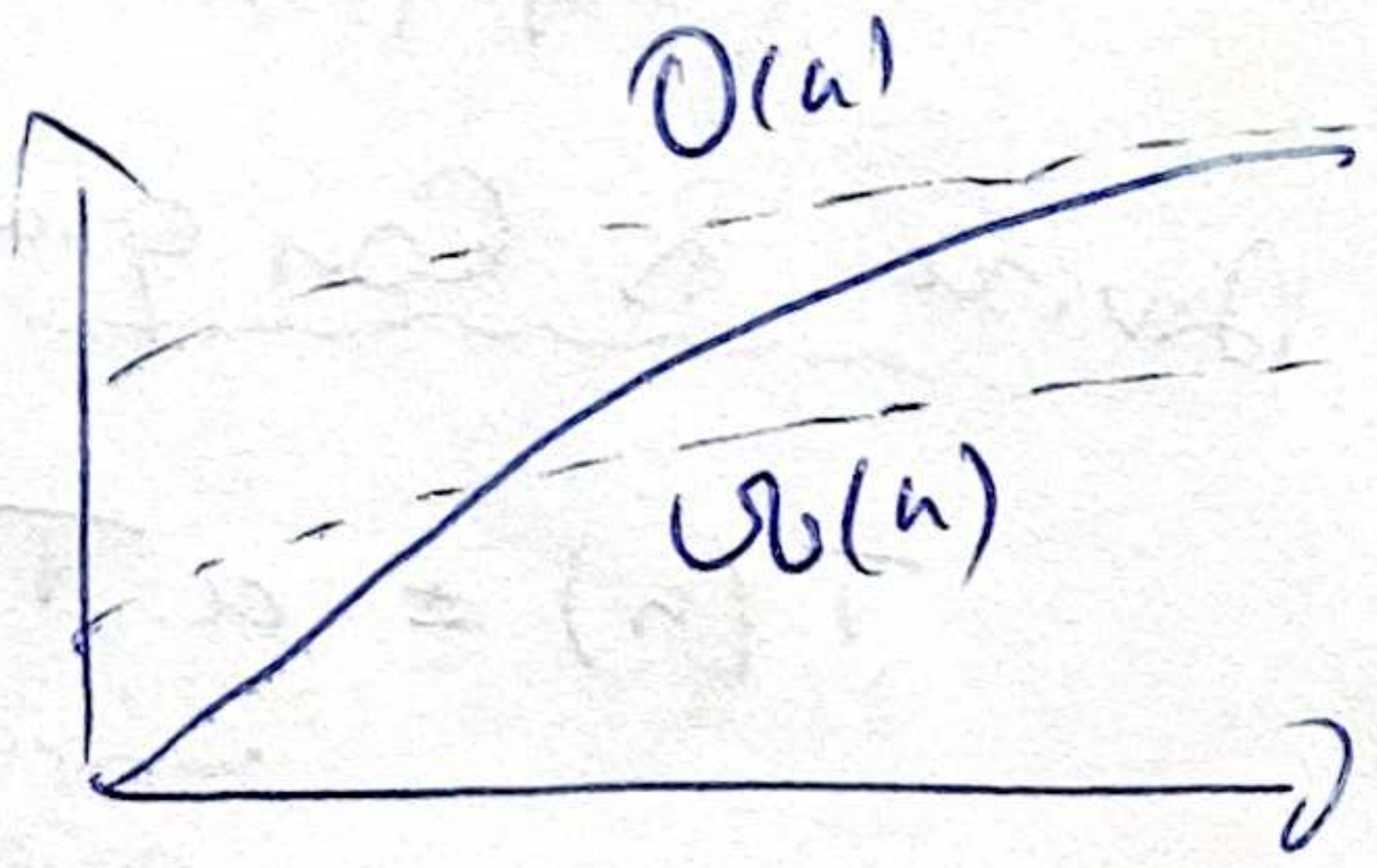
ω $f(n)$ is $\omega(g(n))$

$\lim_{n \to \infty} \frac{g(n)}{f(n)} = 0$



Θ: f is $\Theta(g)$ if

f is $O(g)$ and f is $\Omega(g)$



ex Alg runs in $\Theta(n^2)$ time

→ always in $O(n^2)$ time

→ ∃ inputs ∀ large enough n taking $\Omega(n^2)$ time

ex If $f_1(n)$ is $O(g_1(n))$ and
$f_2(n)$ is $O(g_2(n))$ then
$f_1(n) + f_2(n)$ is $O(g_1(n) + g_2(n))$

Proof:

$\exists\, c_1, c_2, N_1, N_2$ s.t.

$f_1(n) \leq c_1 g_1(n) \quad \forall n \geq N_1$

$f_2(n) \leq c_2 g_2(n) \quad \forall n \geq N_2$

$C_3 = \max(C_1, C_2) \qquad f_1(n) + f_2(n) \leq c_3 [g_1(n) + g_2(n)]$

$N_3 = \max(N_1, N_2) \qquad \forall n \geq N_3$

#

Recurrence relations

$F(n) = F(n-1) + F(n-2)$

exact $T(n) = T(n-1) + 7n - 3 \quad \rightarrow n^2$

$T(n) = 2T(n-1) + n - 3 \quad \rightarrow 2^n$

Divide & Conquer general form

$$T(n) = \underbrace{a T\left(\frac{n}{b}\right)}_{\text{a pieces of size } \frac{n}{b}} + \underbrace{cn^k}_{\text{glue together}}$$

Master Theorem

$T(n) = a\,T\left(\frac{n}{b}\right) + cn^k, \quad a \geq 1,\ b > 1,\ c, k \geq 0$

$T(n) = $
① $O(n^{\log_b a})$ $a > b^k$, lots of subproblems
② $O(n^k \log n)$ $a = b^k$, balance out
levels of recursion → each level contributes to gluing together
③ $O(n^k)$ $a < b^k$, not many subproblems, most time spent gluing together
→ first term does not matter

$T(n) = a\,T(n/b)$ ← last term does not matter

$a^2\,T(\frac{n}{b^2})$ $\quad a^{\log_b n} = n^{\log_b a}$

Merge sort

queue is another form

← stack is one form of the list

List : ordered sequence of elements (array or linked list)

$q : [x_1\ x_2 \dots x_n]$

x_1 = head $\qquad q \circ r$ = concatenation of lists

x_n = tail

$n = |q|$

stack op
- head (q) → return (x_1)
- push (q, x) $\quad q := [x] \circ q$
- pop (q) $\quad q := [x_2 \dots x_n]$ return (x_1)

queue ops
- inject (q, x) $\quad q := q \circ [x]$
- eject (q) $\quad q := [x_1 \dots x_{n-1}]$ return (x_n)

assume these ops can be done in 1 step

merge (s, t) ← sorted lists

```
merge (s, t)
  if s = [] return t
  else if t = [] return s
  else if $s_1 \le t_1$ then u := pop (s)
                        else u := pop (t)
  return push (u, merge (s, t))
```

$O(|s| + |t|)$

iterative

```
mergesort (s)
  list q = []
  for x ∈ s
      inject (q, [x])
  while size(q) ≥ 2
      u := pop(q)
      v := pop(q)
      inject (q, merge (u, v))
  if size(q) = 1 return q(1)
```

10 5 9 11 3 8 4 13 1 6 2 12 16 14 7

[10] [5] [9] .... [7]

loop invariant:
input lists are sorted
output list is sorted

[9] [11] ..... [7] [5,10]

[3] [8] ..... [7] [5,10] [9,11]

[5,10] [9,11] [3,8] ....

[3,8] ~~ [5,9,10,11]

Phase 1 - list size $1 \to 2$ $O(n)$ touch each elt once

2 ... $2 \to 4$ $O(n)$

3 $4 \to 8$ $O(n)$

:

$\log_2 n$ $O(n)$ work per phase

$O(n \log n)$

if $|S|$ is not power of 2, pad with 0s

⇒ each phase will add a constant additional work

⇒ difference between power of 2 and not is a constant fact

recursive

```
mergesort(s)
    if size(s)=1 return(s)
    split(s, s1, s2)
    s1 = mergesort(s1)
    s2 = mergesort(s2)
    merge(s1, s2)
```

correctness proof by induction

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$

↑ constant don't matter

$$T(1) = 1$$

$a=2, b=2, k=1$

$a = b^k \Rightarrow O(n \log n)$ for master theorem

more exact:

let $n = 2^k$, $T(n) = 2T\left(\frac{n}{2}\right) + n$

$a n \log_2 n + bn + c$

when not $n = 2^k$, $T(n) = T\left(\left\lceil \frac{n}{2} \right\rceil\right) + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right)$

Graphs + Modeling

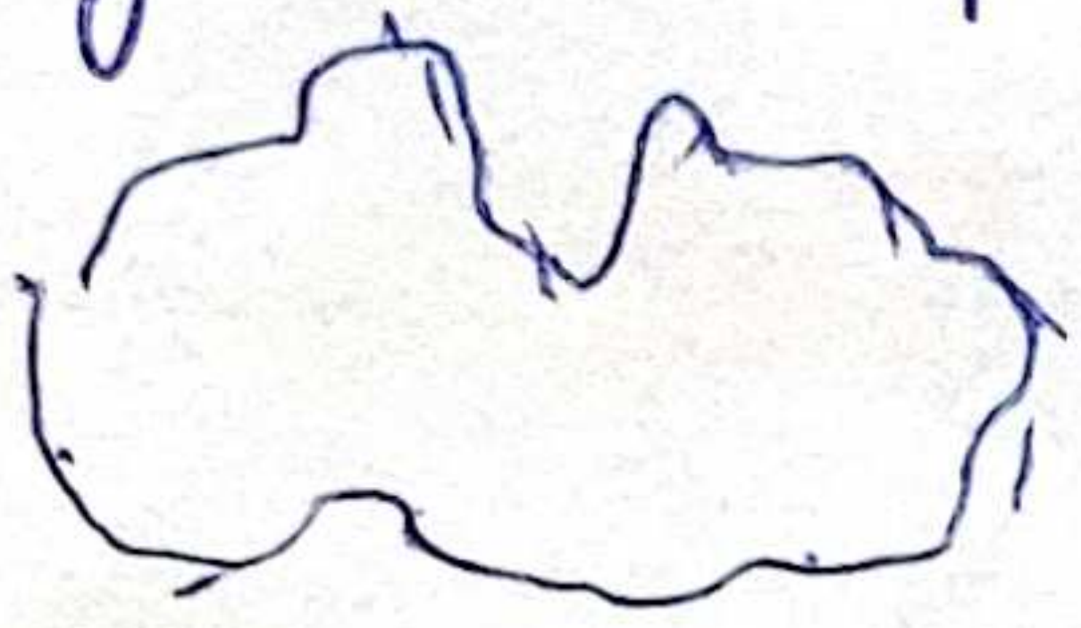
$G = (V, E)$ directed, undirected

vertices edges (u, v) $(u,v) \in E \Leftrightarrow (v,u) \in E$

$E \subseteq V \times V$ $u \to v$ ↔

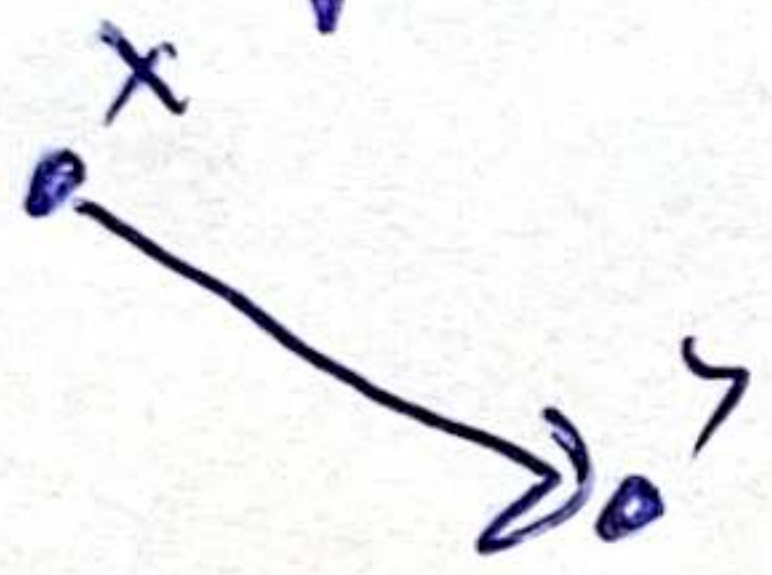
examples

- TSP (traveling salesperson problem)



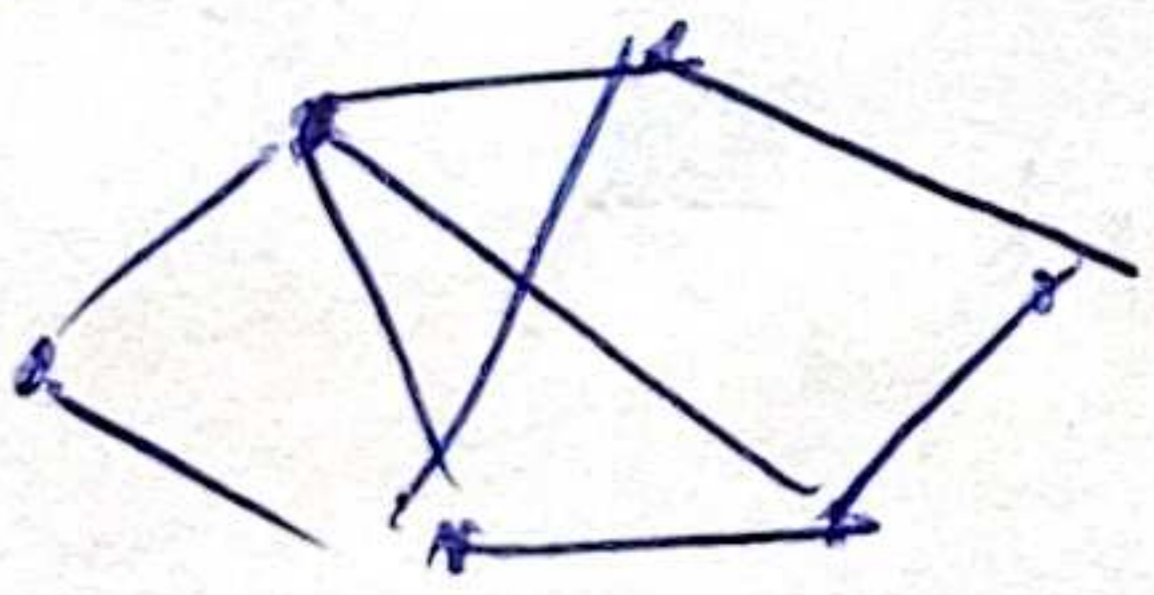
$w: E \to \mathbb{R}$
weight function

- Degrees of separation



x is friends with y
longest shortest path

- Konigsberg bridge
- Coloring problems (maps, register allocation)



colors

$c : V \to \{0, 1, 2, \ldots k\}$
restriction, no two adjacent vertices can have the same color.