CSE101 Final

[redacted]

Runtime

$\lim_{n\to\infty} \frac{f}{g} = \begin{cases} 0 & , f(n) \in O(g(n)) \\ c & , f(n) \in \Omega(g(n)) \\ c>0 & , f(n) \in \Theta(g(n)) \end{cases}$; $g(n) \in O(f(n))$; Both O and Ω

Graphs

adjacency { matrix: $n=|V|$, $n \times n$ array $O(n^2)$ space. constant time check
list: $O(|E|)$, symmetry sort. $\Rightarrow$ $M < |E| \leq |V^2|$ (list / matrix)

- reachability: GraphSearch (G, vertex):
  Initialize $X = \emptyset$. $F = \{s\}$. $U = V - F$ $O(n) \sim O(|V|)$
  while $(F \neq \emptyset)$
  pick v in F
  for each neighbor u of v $O(\sum \deg(v))$
  if $u \notin X$ or F; move u from U to F
  move v from F to X.
  return X. total RT: $O(|V|+|E|)$
- max hand width: add edges for H to L, until $(s,v) \Rightarrow (E_0 < |E|$, $O(|V|^2)$ $< N^2$.
  Binary search: $\log(|V|)(|E|+|V|)$
- Edge Type { tree: solid edge in DFS
  back: leads to ancestor [dir]. In G but not in DFS (node)
  forward: descendent
  cross: neither des. ancestor

- DFS: run explore on all vertices until all explore. $O(|V|+|E|)$
  explore (directed graph, s)
  initialize F as stack
  push (s, F) [prev(s)]
  $pre(s) = \text{null}$
  while $|F| \neq \emptyset$: $v = \text{head}(F)$: if visited (v) pop(v)
  for each neighbor u of v
  if u visited u: pop(u, F); $pre(u) = v$
  visited $(v) = \text{true}$; component $(v) = cc$;
- cycle: Directed cycle iff its DFS output has back edge
  $\Rightarrow$ run DFS, test edge if $(post(u) < post(v)) \Rightarrow$ back edge
- DAG: ① Edge go from higher Post to Lower.
  $\Rightarrow$ linearization: List vertices in decreasing order of post.
  $O(|V|+|E| + |V|\log|V|) \sim$ add+-list $O(|V|+|E|)$
  ② Directed graph is a DAG of its SCC ((u,v) / (v,u))
  { source: no incoming. max post
  sink: no outgoing. min post } $\Rightarrow$ at least one.
  $\Rightarrow$ decomposition: run DFS on G^R; run DFS on G order vertices in decreasing order of post on pre step. ttlc.

Shortest Path

$O(|V|+|E|)$: fine undo.

- BFS: keep track of distance by queue.
  procedure BFS (G, s)
  initialization: for all vertices v in G, $dist(v) = \infty$ $O(|V|)$
  $dist(s) = 0$ $Q = [s]$
  while $(Q \neq \phi)$
  u = eject (Q) $\sum_{\text{adjs}} = O(|E|)$
  for all edges (u,v) in E
  if $dist(v) = \infty$: inject(Q, v); $dist(v) = dist(u)+1$
  runtime: $O(|V|+|E|)$

Dijkstra: weighted. (high level)
$\Rightarrow$ All v in F: $dist(v) = \infty$, $pre(v) = \text{null}$
set $dist(s) = 0$, move s from F to X; $H = \text{makequeue}(V)$
while $(F \neq \emptyset$ and edge $V \times$ to $F)$:
u = deletemin (H)
for all edges (u,v) in E:
if $dist(u) > dist(u) + l(u,v)$: $pre(v) = u$; decreasekey (H, v)
- runtime: $|V| \times$ deletemin + $(|V|+|E|) \times$ decreasekey / insert
  makequeue + deletemin $\times |V|$ + decreasekey $\times |E|$

Priority Queue

- Binary heap: key(parent) < key(child); rebalance each deletemin
  each level from left to right.

	makequeue	deletemin	decreasekey	Dij RT
Array	$\|V\|$	$O(\|V\|)$	$O(1)$	$O(\|V\|^2)$
Binary H	$\|V\|$	$O(\log\|V\|)$	$O(\log\|V\|)$	$O(\|V\|+\|E\| > \log\|V\|)$
d-ary	~~$d\log(V)$~~ / ~~$\log d$~~	$O(\frac{\log V}{\log d})$	$O(\frac{\log V}{\log d})$	$O(\|V\| \cdot d + \|E\|) \cdot \frac{\log\|V\|}{\log d}$
Fib	~~$d+\log(V)$~~	$O(\log V)$	$O(1)$	$O(\|V\|\log\|V\| + \|E\|)$

BH Array

* { Dense: $|E| = |V|^2 \Rightarrow |V^2|\log|V|$
Sparse: $(|E|+|V|)\log(V)$ $O(|V|^2)$

Ex: integer $1, 2, \dots W-1$. $\Rightarrow$ PQ $V(W-1)$

MST: spanning tree: subgraph of Undir G st' G' tree, all vertices in $G(V,E)$ are connected.
~ MST: minimize the total weight of the graph.
$\Rightarrow$ it's a tree: Acyclic, connected, undirected.

- properties of tree
  { 1. remove any edge disconnect the graph [if cycle, x disconnect]
  2. n vertices. A tree iff $(n-1)$ edges
  3. unique path between nodes.

- cut property:
  Suppose edge X part of a MST of $G=(V,E)$. Pick any subset nodes S for which X doesn't cross between S and $V-S$. Let e be the lightest across. $X \cup \{e\}$ is part of some MST.

- Kruskal's
  $\Rightarrow$ procedure kruskal's (G, w)
  for all $v \in V$: makeset (v).
  $X = \{\}$
  sort edges E in increasing order by weight
  for all edges $(u,v) \in E$ until $|X| = |V|-1$ (connected)
  if find $(u) \neq$ find (v): union (u,v), add (u,v) to X.
  $\Rightarrow$ Runtime: $|V| \cdot$ makeset + $2|E| \cdot$ find + $(|V|-1)$ union + sort$(|E|)$

Disjoint Set

{ makeset: put element of S into set
find: return name of subset
union: union set of u and set of v

① Array: leader(u) = index of its head
② Up-tree: directed tree with ranks (child → parent > rank = height)
{ Any root of k has at least 2^k vertices
n vertices, at most $\frac{n}{2^k}$ vertices of rank k
maximum rank is $\log(n)$
③ Path compression: $\log^*(n)$

	makeset	find	union	Kruskal's
Array	$O(1)$	$O(1)$	$O(\|V\|)$	$O(\|V\|^2)$
uptree	$O(1)$	$O(\log(n))$	$O(\log n)$	$O(\|E\|\log\|V\|)$
path	$O(1)$	$O((\log^*(n)-1))$	$O(\log^*(n)-1)$	$O(\|V\|+\|E\| + \|E\|\log^*\|V\|)$

- Prim's
  $\Rightarrow$ procedure Prim's
  move s from F to X
  find min $\{w(u,v)\}$ among all edges such that u in X, v in F
  move v from F to X.
  - next lightest connecting tree with the rest
  - runtime same as Dijkstra's.

Property of MST.
lightest edge e { unique $\Rightarrow$ part of every MST
any $\Rightarrow$ part of some MST.
(w/ adjacent u v)

palindrome: $i = (n-1)$ to 1; for $j = i+1 \cdots n$:
$P(i,j) = \begin{cases} x=y & \sim i+1, j-1 \\ \max(P(i,j-1), P(i+1,j)) & \end{cases}$ $\Omega(n^2)$

library: $L[i] = \min_{1 \leq k \leq j} [\max_{i-l+1 \leq k \leq j} (h_k) + 2(i-l)]$

Greedy Algorithm

- Format:
  - instance: input — constraint: output's property to count as solution
  - solution format: output — objective function: quantity max/min.

Modify the solution:

- claim: Let g_1 be first greedy choice. Let OS be any other solution that meets all requirements and does not include g_1. Then there is a solution OS' that includes g_1 and meet all constrains that is at least as good as OS.
- Exchange argument:
  - g_1 meet requirement for first greedy choice
  - OS meet requirement for the problem $\Rightarrow$ Define OS' in term of g_1 and OS tricky.
- Show: OS' is valid solution
  - compare objective function of OS and OS' ~ OS' same/better.
- Prove by induction: for any input of size $n \geq 1$, the greedy strategy kg
  1. Base case: $n = 1/0$. greedy choice $\Rightarrow$ optimal solution
  2. IH. for some $n \geq 1$. the greedy choice is optimal for all $1 \leq k < n$.
  3. Inductive step:

$$T(OS) \underset{OATS}{\overset{IS}{\geq}} T(OS') = T(I g_1) + S(I') \geq T(I g_1) + \overset{GS(2)}{GS}[I'] \quad \text{IH.}$$

- Greedy stays ahead:
  - Define progress $\rightarrow$ order OS $\rightarrow$ induction show GS each step $\geq$ OS ~~$\Rightarrow$ GS better~~.
  - * Assume by contradiction OS strictly better than GS $\Rightarrow$ progress argument contr
- Achieve the bnd: $cost(GS) \leq bond \leq cost(OS)$
- Greedy Approximation: $cost(GS) \leq k\ cost(OS)$; $value(GS) \geq value(OS)/c$

Divide and Conquer

$$S_n = \frac{a_1(1-r^n)}{1-r}$$

- Master Theorem:

$T(n) = aT(\frac{n}{b}) + O(n^d)$ for some const $a > 0$, $b > 1$, $d \geq 0$.

$$T(n) = \begin{cases} O(n^d) & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a \end{cases}$$

$a^{\log_b n} = n^{\log_b a}$ (tree: n, $\frac{n}{b}$, depth $\log_b n$)

k-th level: a^k subproblems. each size of $\frac{n}{b^k}$ $= a^k \times O(\frac{n}{b^k})^d = O(n^d) \times (\frac{a}{b^d})^k \Rightarrow$ geometric series: $T(n) = O(n^d \sum_{k=0}^{\log_b n} (\frac{a}{b^d})^k)$

- Multiplication:

$(a + bx)(c + dy) = ac + ady + bcx + bdy$

$\Rightarrow x = [x_L][x_R]$ $y = [y_L][y_R]$.

$R_1 = mks(x_L, y_L)$, $R_2 = mks(x_R, y_R)$, $R_3 = (x_L + x_R)(y_L + y_R)$

return $R_1 * 2^n + (R_3 - R_1 - R_2) 2^{\frac{n}{2}} + R_2$.

- runtime: $T(n) = 3T(\frac{n}{2}) + O(n) \sim n^{1.58}$ $\Rightarrow$ 3 poly $n^{1.43}$
- $\Rightarrow$ cook-tom: divide into k-subproblems, combine with 2k-1 multi

$T(n) = (2k-1) T(\frac{n}{k}) + O(n) \Rightarrow T(n) = O(n^{\log_k 2k-1})$

- FFT

	coeff	root	sample
eval	$O(n)$	$O(n)$	$O(n^2)$
Add	$O(n)$	∞	$O(n)$
Mult	$O(n^2)$	$O(n)$	$O(n)$

$O(n \log n)$

Given coefficient $(a_0, a_1 \cdots a_{n-1})$

n root unity: $X = \{w_n, w_n^2, \cdots w_n^n\}$ collapsing set

$A_e = \langle a_0, a_2 \cdots a_{n-2} \rangle$, $A_o = \langle a_1, a_3 \cdots a_{n-1} \rangle$

compute $A_e(\partial)$ and $A_o(\partial)$ for all $\partial \in X^2$

$A(x) = A_e(x^2) + xA_o(x^2)$ for all $x \in X$

$T(n, |X|) = 2T(n/2, |X^2|) + O(n + |X|) \Rightarrow T(n, |X|) = 2(n/2, |X|/2) + O(n + |X|)$

- Search: $\Omega(\log(n)) \Rightarrow$ Binary search
- Sort: $\Omega(\log(n!)) \Rightarrow \Theta(n \log n)$ $[\int_1^n \log x\, dx]$. Must take $O(n^2)$
  - $\Rightarrow$ mergesort: if $n > 1$, recursively sort 2 subparts; else return a

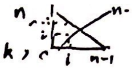
$T(n) = 2T(\frac{n}{2}) + O(n)$ merge runs $O(n \log n)$

- $\Rightarrow$ quicksort: if $n \leq 1$ return a.
  - $O(n \log n) \sim O(n^2)$: set v to random in a; partition S_L, S_v, S_R; return quicksort(S_L) · S_v · quicksort(S_R)

$$ET(n) = \frac{1}{n} \left(\sum_{i=1}^{n} ET(n-i) + ET(i) + O(n) \right) = \frac{2}{n} \sum_{i=1}^{n} ET(i) + O(n) = O(n \log n)$$

($|S_R|$, $|S_L|$)

- Selection: $\Omega(n)$



Runtime: $O(n) \sim O(n^2)$

$\Rightarrow$ select$(s, k) = \begin{cases} \text{select}(S_L, k) \\ v \\ \text{select}(S_R, k - |S_v| - |S_L|) \end{cases}$

$ET(n) = \frac{1}{n} \sum_{i=1}^{n} ET(\max(i, n-i)) + O(n)$.

$ET(n) \leq ET(\frac{3n}{4}) + O(n) \Rightarrow \leq O(n)$; $ET(n) \leq \frac{1}{2} ET(\frac{3}{4}n) + \frac{1}{2} ET(n) + O(n)$

$\Rightarrow$ deterministic: ① split into 5 sets, find medians; find median of median using recursive $T(n/5)$

② quick select

2) less than 10, sort return k; Partition L in S(i) of 5 elements;

$m[i] = \text{Median}(B[i])$ $O(n)$; $\text{MofM}(m[\cdots], m[n/5]], n/10)$. $T(\frac{n}{5})$

split the set S_L, S_m, S_R $\Rightarrow$ quick select $T(\frac{7n}{10})$

$T(n) = T(\frac{n}{5}) + T(\frac{7n}{10}) + O(n) \sim O(n)$. induct $T(n) < cn$

- BinaryHeap: recurse on $(0, k_1)$, two of $2^k - 1$: $T(n) = 2T(\frac{n}{2}) + O(\log n)$

$L(n) < T(n) < U(n) \Rightarrow T(n) \in \Theta(n)$

- uneven runtime: $T(n) = T(L) + T(R) + O(LR)$, $T(n) \leq cn^2$ for $n \geq 1$.

$c > T(1)$ and $> O(LR)$. $T(k) \leq ck$ for all $k < 0$.

$T(n) \leq cL^2 + cR^2 + cLR < c(L+R)^2 = c(n-1)^2 < cn^2$.

- greatest overlap

$\Rightarrow$ procedure (sort$(a_1 b_1 \cdots a_n b_n)$)
if $n = 1$ return 0.
~~overlap1~~ = overlap side (LS)
olap2 = overlap side (RS)
overlapbetween (LR, RS)
return max.

$\Rightarrow$ procedure overlap between $(a_1 b_1 \cdots c_1 d_1 \cdots)$
if $k == 0$ | $l == 0$ return 0
find latest red interval
compare it to all blue ones. $O(n)$

runtime: $T(n) = 2T(\frac{n}{2}) + O(n)$

Dynamic Programming

- maximal independent set

MIS3 $(G = (V, E))$
if $|V| = 0$ return $\emptyset$;
pick vertex v: $S_1 = v + MIS(G - v - N(v))$ if $\deg(v) = 0/1$, return S_1
$S_2 = MIS3(G - v)$ return max S_1, S_2.

$T(n) = T(n-1) + T(n-3) + C \sim O(2^{0.6n})$

- k-shot strategy: $p(1) \cdots p(n)$: price for stock n days.

m pair of $(b_1, s_1) \cdots (b_m, s_m)$; $\max \sum_{1 \leq i \leq m} p(s_i) - p(b_i)$

1. Define the subproblems and array:
   Let $MP(i, j)$ = max return among all j-shot strategy for i days.
2. Base cases: $MP(i, 0)$: never buy or sell $MP(1, j) = 0$. x buy sell same
3. the recursion used to fill the array:
   $MP(i, j) = \max\{MP(i-1, j), \max_{1 \leq h \leq i-1} \{(p(i) - p(h)) + MP(h-1, j-1)\}$
4. ordering of sub-problem:
   cell (i, j) depend on $(i, j-1)$ and $(h-1, j-1)$ for $1 \leq h \leq i-1$.
   for $i = 2 \cdots n$
   for $j = 1 \cdots n$.
5. Final for of output: return $MP(n, k)$
6. Pseudocode: $\Rightarrow$ kshotprofit$(p(1) \cdots p(n); k)$
   $MP(i, 0) = MP(0, j) = MP(1, j) = 0$
   for $(i = 2 \cdots n)$
   for $(j = 1 \cdots k)$: $MP(i, j) = \max\{MP(i-1, j), \max\{p(i) - p(h) \cdots\}$
   return $MP(n, k)$
   runtime: $O(nk) \times O(n) = O(n^2 k)$

infinite return: $B(i, n-1) \neq B(i, n)$

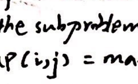
BellFord: $BF(i, t) = 0$
$BF(i, v) = \infty$
for $i = 1 \cdots n$, $i = 2 \cdots n$
$BF(i, t) = \min$
for $(i = 2 \cdots n)$
if $BF(i, n-1) \neq BF(i, n)$ negative
$(BF(i, t-1)$
$+ w(V_j, t)$

PNP: A decision problem is an algorithm takes input an instance I and solution S and determines if S is a valid solution of I in polytime of $|I|$.

{ set of all decision problems NP; polynomial time P }

{ NP-hard & NP; NP complete reduce to it }

- Graph path:



- no bijection on G
- no bijection on G^n
- Greedy { cross with v_j $\Rightarrow$ add; x cross with v_j $\Rightarrow$ x possible }

- Binary. Put first elements of each list; repeat: pop min, replace heart smallest in the list $\Rightarrow O(n \log k)$
- 3SAT problem
- $S(n) = S(L) + S(R) + C$; $n > 1$, $S \leq cn$ (Assume $S(i) \leq c$)

DP: $D[0, j] = D[i, 0] = 0$; $D[i, j] = D[i-1, j]$; if include u_i:

$D[i, j] = \max_{(u_i, v_k) \in E} \{1 + D[i-1, k-1]\}$.

MaxMatch
Initialize: $D[i, 0]$ $D[0, j]$ (1+r)
for $i = 1 \cdots l$, $j = 1 \cdots r$.
$\max = D[i-1, j]$
for $(u, k \in E)$
$1 + D[i-1, k-1]$

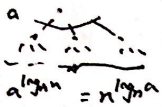
$O(\sum_{i=1}^{l} r \deg u_i) = O(mn)$

Master Theorem

- $T(n) = aT(\frac{n}{b}) + O(n^d)$ for some constant $a>0$, $b>1$, $d\geq 0$, then:

$$T(n) = \begin{cases} O(n^d) & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a \end{cases}$$

- proof idea:



k-th level: a^k subproblems, each of size $n/b^k = a^k \times O(\frac{n}{b^k})^d = O(n^d) \times (\frac{a}{b^d})^k$. ⇒ geometric series: $T(n) = O(n^d \sum_{k=0}^{\log_b n} (\frac{a}{b^d})^k)$

If $T(n) = aT(n-b) + O(n^d)$ $a>1$, exponential time

Sorting

- binary search tree of n! possible sequence leaves. ⇒ $\log(n!)$

① mergesort (a[1...n]). $\log(n!) = \sum_{k=1}^{n} \log k = \int = n\log n - n + 1$

if n>1: return merge(mergesort(a[1...n/2]), mergesort(a[n/2+1...n]))
else return a.

- merge(x, y): compare the first term then recursively merge rest.
- $T(n) = 2T(\frac{n}{2}) + O(n) \Rightarrow O(n\log n)$

② Quicksort (a[1...n])

- if n≤1: return a
  set v to be a random element, partition into S_L, S_V, S_R.
  return quicksort(S_L) S_V quicksort(S_R).
- expected $O(n\log n)$

③ other sorting ~ $O(n^2)$

selection

- all selection takes $O(n)$
- $\text{selection}(S,k) = \begin{cases} \text{selection}(S_L, k) & \text{if } k \leq |S_L| \\ v & \text{if } |S_L| < k \leq |S_L| + |S_V| \\ \text{selection}(S_R, k - |S_L| - |S_V|) & \text{if } k > |S_L| + |S_V| \end{cases}$
- runtime $\begin{cases} T(n) = T(n/2) + O(n) & \text{best case } O(n) \\ T(n) = T(n-1) + O(n) & \text{worst case } O(n^2) \end{cases}$ ← closest

Multiplication

$(a+bx)(c+dx) = ac + adx + bcx + bdx^2$.

- multiply:

$R_1 = \text{multiply}(a_L, c_L)$ $R_3 = \text{multiply}(a_L + a_R)(c_L + c_R))$

$R_2 = \text{multiply}(a_R, c_R)$ return: $R_1 * 2^n + (R_3 - R_1 - R_2) 2^{\frac{n}{2}} + R_2$

- runtime: $T(n) = 3T(\frac{n}{2}) + O(n) \sim n^{1.58}$; 3 poly ~ $n^{1.45}$.

cook toom: divide into k subproblems.

$T(n) = (2k-1)T(\frac{n}{k}) + O(n) \Rightarrow T(n) = O(n^{\log_k(2k-1)})$

FFT.

	Coeff	Roots	Simple
Eval	$O(n)$	$O(n)$	$O(n^2)$
Hold	$O(n)$	$O(\infty)$	$O(n)$
Multi	$O(n^2)$	$O(n)$	$O(n)$

$O(n\log n)$

* deterministic selection: split set into 5. find medians.
recursive find median $\to T(n/5)$; partition the set, recurse

```
main(L,k)
  if L < 10 return k-th.
  Find medians of 5 subset.
  m = main(m(1) ... m(n/5)}, n/10).
  split the list, ⇒ quick select.
```

$T(n) = T(\frac{n}{5}) + T(\frac{7n}{10}) + O(n)$

$T(n) \leq cn \Rightarrow T(n) \leq cn$.

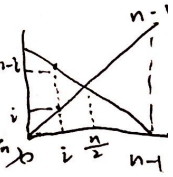
* Linear time selection

Examples:

- runtime:
  ① $T(n) = 3T(n/2) + O(n\log n)$. $n < n\log n < n^2$.
  ② $T(n) = T(L) + T(R) + O(LR)$: $T(n) \leq cn^2$ for $n \geq 1$.
  $c > T(1)$ and $> O(LR)$. $T(k) \leq ck$ for all $k < n$.
  $T(n) \leq cL^2 + cR^2 + cLR < c(L+R)^2 = c(n-1)^2 < cn^2$.
- Binary heap: insert ~ $n\log(n)$. ⇒ $O(n)$
  ... leave root, break into 2 parts, $T(n) = 2T(\frac{n}{2}) + 1$
- Maximum contiguous subarray.
  MaxContSum(L), MCS(R), AcrossSum(L,R).
- Fence: leave all routes $[R.a, R.b] \times [R.a, R.b]$.
  recurse on two subproblems. ⇒ ~~compare~~
  combine possible results ~ compute min.
- Power of Two.
  P.T(n) if n=0 return 1; If n=1 return 2 $P = \frac{n}{2}$ even $P = 2^{(n-1)/2}$ o
  $P = P_1 T(\frac{n}{2})$; $P = \text{KSMult}(P, P)$.
  If n mod 2 = 1 then $P = \text{add}(P, P)$ return P.
- max subset sum
  n=0, return 0, n=1, retur V[1]. n=2 return max[V(1) V(2)
  n=3 return max(V(1), V(2) V(3), V(1)+V(3)
  w = V[mid] + maxSm ....; w_2 = maxSm[V(mid-1)
  $T(n) \leq 4T(n/2) + O(n)$
- Expected:

$$ET(n) = \frac{1}{n}\sum_{i=1}^{n} ET(\max(i, n-i)) + O(n)$$

$P[\frac{n}{4} \sim \frac{3n}{4}]$: $ET(n) \leq \frac{1}{2}ET(\frac{3n}{4}) + \frac{1}{2}T(n) + cn$



$ET(n) \leq ET(\frac{3n}{4}) + O(n) \Rightarrow$ linear ti

- convert to binary: to decimal
  split to L and R, recursively convert L and R to decim
  compute power of Two $(\frac{n}{2})$ ⇒ $2^{n/2}$; multiply with L, a
  $L \cdot 2^{\frac{n}{2}} + R$.
  ∴ $T(n) = 2T(\frac{n}{2}) + O(n^{1.58})$
- runtime: $T(n) = T(L) + T(R) + c$.
  - constant $d > c$, $d > T(1)$ and $T(n) \leq dn$ fo
  - Base case $T(1) < d$. ⇒ Induction $T(k) \leq dk$ for
    $T(n) \leq dL + dR + c \leq d(L+R+1) \leq dn$

CSE 101 [redacted] Qw22

Greedy Algorithm

Format

instance: input — constraint: output's property to count as solution
solution format: output — objective function: quantity we're max/min.

Method 1 - Modify solution (recursion)

- claim: Let g_1 be the first greedy choice. Let OS be any other solution that meet all requirements and does not include g_1. then there is a solution OS' that includes g_1 and meet all constraints and is at least as good as OS.
- Exchange argument:
  g_1 meet requirm for first greedy choice ⇒ Define OS' in terms of OS meet requirm for the problem g_1 and OS to include g_1.
- Show: OS' is valid solution.
  compare objective function of S and OS' → OS' same/better.
- Prove by induction: for any input of size $n \geq 1$, the greedy strategy is optimal
  ① Base case: $n=1$ / 0. greedy choices ⇒ optimal solution.
  ② IH. for some $n \geq 1$. the greedy is optimal for all $1 \leq k < n$.
  ③ Inductive step: $G(S(I))$.
  $T(OS) \overset{BT}{\geq} T(OS) = T([g_1] + S(I')) \overset{IH}{\geq} T([g_1] + GS(I'))$

Method 2 - Greedy stays ahead (iterative modify the solution)

- Iter MTS: Let $g_1, g_2, \dots g_r$ be decisions made in order by greedy strategy. For each $0 \leq i < r$. There is an optimal solution OS_i that includes $g_1, g_2, \dots g_i$.
- Prove by induction:
  ① Base case: For $i=0$, we let OS_0 by any optimal solution. x agree any greedy decisions.
  ② Assume there is an optimal solution OS_{i-1} that includes $g_1, g_2 \cdots g_{i-1}$. If also includes g_i, $OS_i = OS_{i-1}$. Or.
  Define $OS_i = g_1 \cdots g_{i-1}$ + i'th move of OS_{i-1} to g_i.
  prove OS_i meets requirements.
  compare $|OS_i|$ and $|OS_{i-1}|$.

Minimum Spanning Tree

- Given graph $G=(V,E)$, MST is a Tree $T=(V,E')$ that minimizes total weight of T. Acyclic, connected.
- properties:
  1. remove a cycle edge x disconnect a graph.
  2. A tree on node ⟷ n-1 edges (connected) undirected
  3. An undirected graph is a tree iff unique path between any pair
- cut property:
  Suppose edges X are part of a MST of $G=(V,E)$. Pick any subset of nodes S for which X doesn't cross between S and $\bar{S}$.
  Let e be the lightest edge across. $X \cup \{e\}$ is part of some MST.
- union/find:
  - make root of smaller tree point to root of larger tree.
  - properties: rank(x) < rank(p(x)).
    A node of rank k has at least 2^k descendants.
    Rank k+1 ← union k and k.
    n elements, at most $n/2^k$ nodes of rank k.
  - runtime: union/find ~ $O(\log(n))$,
  - path compression: reduce runtime to $O(1)$.

- Kruskal's algorithm:
  sort edges
  $T = \{\}$
  repeat until n-1 edges
  add next edge which don't form a cycle
  return T.

```
For all u ∈ V:
    makeset(u);
X: {}
sort edges E by weight:
For all edges (u,v) belongs to E, in increasing order of weight
    if find(u) ≠ find(v):
        Add edge (u,v) to X
        union(u,v)
return X;
```

- Prim's algorithm:
  $X = \{\}$
  repeat until $|X| = |V| - 1$
  pick a subset of V's for which X has no edge between S and V−S. Let e be the min edge between
  $X = X \cup \{e\}$
  return X.

Ex: MST ⇒ weight w ⇒ all MST h
origin w.

Dijkstra Algorithm - shortest path

```
For all u ∈ V:
    dist(u) = ∞; pre(u) = null;
dist(S) = 0
H = makequeue(V) (using dist values)
while (H is not empty).
    v = deletemin(H)
    for all edges (u,v) belongs to E:
        if dist(v) > dist(u) + l(u,v)
            prev(v) = u
            decreasekey(H,v)
```

- runtime: priority queue.

	deletemin	insert/decreasekey	$\lvert V\rvert \times$ deletemin + $(\lvert V\rvert+\lvert E\rvert) \times$ inser
Array	$O(\lvert V\rvert)$	$O(1)$	$O(\lvert V\rvert^2)$
Binary Heap	$O(\log\lvert V\rvert)$	$O(\log\lvert V\rvert)$	$O((\lvert V\rvert+\lvert E\rvert)\log\lvert V\rvert)$
d-ary heap	$O(\frac{d\log\lvert V\rvert}{\log d})$	$O(\frac{\log\lvert V\rvert}{\log d})$	$O(\lvert V\rvert \cdot d + \lvert E\rvert)\frac{\log\lvert V\rvert}{\log d})$
Fibo	$O(\log\lvert V\rvert)$	$O(1)$ (amortized)	$O(\lvert V\rvert \log\lvert V\rvert + \lvert E\rvert)$

- Binary heap: { each level filled from left to right; key(parent) < children.
  v_1: repeat
  v_2: each one } ⇒ h pro

Ex: Proj$_i$, K_i, Co. ⇒ max profit; k projects
V_1: $OS' = OS$ inorder + project$_1$.
v_2/v_3: ① x proj$_j$, ⇒ same as v_1
② v purge at j'th, switch the order ⇒ meet requ on n
v_3: one, x limit
induction

Ex: in complete: $n^2 = |E|$. $\frac{n(n-1)}{2}$; complete bipartite $\frac{n}{2}$: $(\frac{n}{2})(\frac{n}{2}) = n^2 = |E|$
$\sqrt{n} \times \sqrt{n}$: 4 corner ~ degree 2; $4(\sqrt{n}-2)$ ~ degree 3; $(\sqrt{n}-2)(\sqrt{n}-2)$ degree 4.
$|E| = \frac{1}{2}[4(2) + 4(3)(\sqrt{n}-2)3 + (\sqrt{n}-2)(\sqrt{n}-2)4]$.
- hypercube $n = 2^k$. each vertex k neighbors; $m = \frac{1}{2}\sum_{v \in V} \deg(v)$, $m = \frac{1}{2}nk$

Ex: ≥ |V|-1 edges ⇒ unique heaviest edge might be in
Ex: e minimum weight ⇒ must be part of MST.
Ex: v → lighter e ⇒ must be part of MST
Ex: lightest edge unique ⇒ part of every MST

Ex: $a_i b_j$ max area; ⇒ $a_i b_j$ $b_i a_j$. $a_i b_i + a_j b_j$ $OS' - OS \geq 0$.

Ex: part first at x_{i+k} eliminate; OS first s; OS' first $x+k = g$.
⇒ g covers all x covers.

Ex: finished — may g with first d of OS = OS'

Ex O_{xm}: { X O_{xn}, O_{xw} pair with any other ⇒ add pair
O_{xn} x pair, O_{xw} pair with O_{xj} ⇒ replace O_{xj} with O_{xn}
O_{xw} x pair ⇒ replace O_{xi} with O_{xw}.
O_{xn} pair with O_{xi} and O_{xw} pair O_{xj} ⇒ replace 2 pairs

case 1: $S_1 + S_n < P$ ⇒ find $x_1 x_2 \cdots x_n$
case 2: $S_1 + S_n \geq P$ ⇒ MST