Jin Wei CS3230

$\sum^n i = \frac{n(n+1)}{2}$

AP $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$ $\quad \sum^n i^2 = \frac{n(n+1)(2n+1)}{6}$

GP $\sum_{i=0}^{n-1} ar^i = \frac{ar^n - a}{r-1}$ $\quad H_n = \sum_{k=1}^{n} \frac{1}{k} = \ln n + O(1)$

$\sum_{i=0}^{\infty} ar^i = \frac{a}{1-r}$ $\quad \sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$

$\binom{n}{r} = \frac{n!}{r!(n-r)!}$

$\binom{n}{2} = \frac{n(n-1)}{2}$

$nP_r = \frac{n!}{(n-r)!}$

height of $\sqrt{n}$ is $\lg\lg n$
→ lowest is $n^{1/2^k}$

$n^{1/2^k} = 2$

$2^{\frac{1}{k}} \lg n = \lg 2$

$\lg n = 2^k$

$k = \lg\lg n$

$(a^m)^n = a^{mn}$ ¦ $a^m a^n = a^{m+n}$ ¦ $e^x \geq 1 + x$ ¦ $\lg(AB) = \lg A + \lg B$

$\lg_a 1 = 0$ ¦ $\lg_a a = 1$ ¦ $a^{\lg_a b} = b$ ¦ $a^{\lg_b c} = c^{\lg_b a}$ ¦ $\lg_b a = \frac{\lg_d a}{\lg_d b}$

$\lg(n!) = \Theta(n \lg n)$ $\quad n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right)$ $\quad (\lg n)! = 2^{\Theta(\lg n \lg\lg n)}$

$\lg\lg n \ll \lg n \ll \lg^2 n \ll \sqrt{n} \ll n^x \ll n^{\lg n} \ll (\lg n)! \ll x^n \ll (\lg n)^n \ll n! \ll n^n$

$O(g(n)) = \{ f(n) : \exists c > 0, n_0 > 0 \text{ st } 0 \leq f(n) \leq c g(n) \ \forall n \geq n_0 \}$

$\Omega(g(n)) = \{ f(n) : \exists c > 0, n_0 > 0 \text{ st } 0 \leq c g(n) \leq f(n) \ \forall n \geq n_0 \}$

$\Theta(g(n)) = \{ f(n) : \exists c_1 > 0, c_2 > 0, n_0 > 0 \text{ st } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \ \forall n \geq n_0 \}$

$o(g(n)) = \{ f(n) : \forall c > 0, \exists n_0 > 0 \text{ st } 0 \leq f(n) < c g(n) \ \forall n \geq n_0 \}$

$\omega(g(n)) = \{ f(n) : \forall c > 0, \exists n_0 > 0 \text{ st } 0 \leq c g(n) < f(n) \ \forall n \geq n_0 \}$

NOTE: $2^{5n} \neq O(2^n)$, trigo is good counter example

L'hopital if num = den = 0 or $\pm\infty$

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

$\nabla = 0 \Rightarrow$ small o

$\nabla < \infty \Rightarrow$ big O

$0 < \nabla < \infty \Rightarrow \Theta$

$\nabla > 0 \Rightarrow \Omega$

$\nabla = \infty \Rightarrow \omega$

degree-k $= O(n^k) = o(n^{k+1}) = \omega(n^{k-1})$

✓ Transitive $f(n) = \nabla(g(n))$ & $g(n) = \nabla(h(n)) \Rightarrow f(n) = \nabla(h(n))$ [all 5]
✓ Reflexive $f(n) = \nabla(f(n))$ [$O\,\Omega\,\Theta$]
✓ Symmetry for Θ [Θ]
✓ Complementarity for (O, Ω), (o, ω)
$f(n) = O(g(n)) \iff g(n) = \Omega(f(n))$

Iterative $\quad i \Rightarrow i+1$

- Correctness : loop invariant
  initialisation, maintainence, termination
- Runtime : obvious

Recursive

- Correctness : Base case + Strong induction (for $< n$)
- Runtime : Recurrence relation

1. Recursion Tree (show $\leq$ & $\geq$ if bounding!)
2. Master Method (CANNOT differ by $\lg n$)

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$

$a \geq 1$, $b > 1$, $n^{\lg_b a}$

Case 1: $f(n) = O(n^{\lg_b a - \varepsilon})$, $\varepsilon > 0$
$f(n)$ grows polynomially slower than $n^{\lg_b a}$ by n^ε [leaf-heavy]
$\Rightarrow \Theta(n^{\lg_b a})$

i.e. $\lg n = O(n^{1-\varepsilon})$ $\quad \frac{n^2}{\lg n} \notin O(n^{2-\varepsilon})$
$n \lg n = O(n^{2-\varepsilon})$ $\quad n \lg\lg n \in O(n^{1-\varepsilon})$

Case 2: $f(n) = \Theta(n^{\lg_b a} \lg^k n)$, $k \geq 0$
$f(n)$ grows at similar rates with $n^{\lg_b a}$
$\Rightarrow \Theta(n^{\lg_b a} \lg^{k+1} n)$
[similar work on each level]

Case 3: $f(n) = \Omega(n^{\lg_b a + \varepsilon})$, $\varepsilon > 0$
$f(n)$ grows polynomially faster than $n^{\lg_b a}$ by n^ε [root-heavy]
AND $f(n)$ satisfies regularity condition
that $af\left(\frac{n}{b}\right) \leq cf(n)$ for some $c < 1$ (state c or range of c)
$\Rightarrow \Theta(f(n))$

i.e. $n^2 \lg\lg n$ CANNOT for n^2
$n \lg\lg n$ CANNOT for n

3. Substition Method
   → guess + induction
   → use more terms
   → use subtraction

Random

Las Vegas : Always correct, random runtime
Monte Carlo : Sometimes incorrect, same runtime

Average running time : for non-randomized algorithms that depend on input
→ need to know distribution of input

Expected running time : for randomized algorithms
→ use $E[X]$

$P(A_k \mid B) = \frac{P(A_k) P(B \mid A_k)}{\sum_{i=1}^{n} P(A_i) P(B \mid A_i)}$

for $n = 2$: $P(A \mid B) = \frac{P(A) P(B \mid A)}{P(A) P(B \mid A) + P(A') P(B \mid A')}$

Bernoulli Trials : p -- success
(geometric distribution) $q = 1 - p$ -- failure

$P_r[X = k] = (1-p)^{k-1} \cdot p$

$E[X] = \sum_{k=1}^{\infty} k \cdot P_r(X = k)$
$= \sum_{k=1}^{\infty} k \cdot (1-p)^{k-1} \cdot p = \frac{1}{p}$

Indicator Random Variable $X_i = \begin{cases} 1, & \text{if } A \text{ occurs} \\ 0, & \text{otherwise} \end{cases}$ $\quad E[X_i] = P_r[A]$

$E[XY] = E[X]E[Y]$ if X and Y are independent

Hashing → CS3230 only cover chaining

(by PHP) for $h : [U] \to [M]$, if $U \geq (N-1)M + 1$, then $\exists$ set of N elements that collide

pairwise indep. $\Rightarrow$ universal, universal $\not\Rightarrow$ pairwise indep.

Universal H if $\forall$ distinct $x, y \in U$: $\Pr_{h \sim H}[h(x) = h(y)] \leq \frac{1}{M}$

Uniform H if $\forall x \in U$, $k \in [M]$: $\Pr_{h \sim H}[h(k) = i] \leq \frac{1}{M}$

let $H = \{h_1, h_2, \ldots, h_n\}$
$h_1(x) = 1$, $h_2(x) = 2$, … (all $x \in U$)
→ NOT universal

pairwise independent H if $\forall$ distinct $x, y \in U$, $i_1, i_2 \in [M]$: $\Pr_{h \sim H}[h(x) = i_1, h(y) = i_2] = \frac{1}{M^2}$

universal:

	a	b
h_1	0	0
h_2	0	1

	a	b
h_1	0	1
h_2	1	0

	a	b
h_1	0	0
h_2	1	0
h_3	0	1

NOT universal:

	a	b
h_1	0	0
h_2	1	1

	a	b	c
h_1	0	0	1
h_2	1	1	0
h_3	1	0	1

for universal:

- expected num of collisions for any N elements $< \frac{N-1}{M}$ $\because$ each other elem $\leq \frac{1}{M}$
- expected cost of n operations is $O(n)$ if $M > N$
- expected num of pairs (i, j) of collisions after adding $x_1, \ldots, x_N$ and $M < N$ is $\leq \underbrace{N \cdot 1}_{\text{self-collision}} + N(N-1) \cdot \frac{1}{M} < 2N$
- expected max. load (max. elem in one slot) $\leq \sqrt{2N} \leq O(\sqrt{N})$

$\because$ total collisions $\geq (\text{max. load})^2$ $\because$ $\binom{\text{max. load}}{2}$ collisions [everything collide in 1 slot]

$\{h_A : A \in \{0,1\}^{m \times u}\}$ is universal where $h(x) = Ax \pmod 2$ (x: col vector)

2-level hashing : $O(1)$ ops, $O(N)$ space

if N_1 = number of X_i and N_2 = number of Y_i

$N_1 = \sum_{i=1}^{n} X_i$ $\quad N_1^2 = \sum_{i=1}^{n} X_i^2 + 2 \sum_{1 \leq i < j \leq n} X_i X_j$ $\quad N_1 N_2 = \sum_{1 \leq i \leq n, 1 \leq j \leq n} X_i Y_j$

Common Recurrences

$T\left(\frac{n}{2}\right) + O(1)$	$O(\lg n)$
$2T\left(\frac{n}{2}\right) + O(1)$	$O(n)$
$T\left(\frac{n}{2}\right) + O(n)$	$O(n)$
$2T\left(\frac{n}{2}\right) + O(n)$	$O(n \lg n)$
$2T\left(\frac{n}{2}\right) + O(n \lg n)$	$O(n \lg^2 n)$
$T(\sqrt{n}) + O(\lg n)$	$O(\lg n)$
$T(n-1) + O(\sqrt{n})$	$O(n\sqrt{n})$
$2T\left(\frac{n}{2}\right) + \frac{n}{\lg n}$	$\Theta(n \lg\lg n)$ (H_n)
$2T(\sqrt{n}) + O(1)$	$\Theta(\lg n)$
$2T(\sqrt{n}) + O(\lg n)$	$\Theta(\lg n \lg\lg n)$

$\int_{i=1}^{x} f(i)\,di \leq \sum_{i=1}^{x} f(i) \leq \int_{i=1}^{x+1} f(i)\,di$

$\left(1 - \frac{1}{n}\right)^n \approx \frac{1}{e}$

$\left(1 + \frac{a}{n}\right)^{bn+c} \approx e^{ab}$

Misc

Comparison-based Sorting
↳ $n!$ universe
↳ cut in half every query
↳ $\Omega(\lg(n!)) = \Omega(n \lg n)$

Amortized Analysis

→ most op cheap
→ rare op expensive

* NO probability

1. Aggregate Analysis
   - → average cost of worst sequence of operations
   - → simple to understand but tedious in practice
2. Accounting (Banker's) Method
   - → set $c(i)$ for each op st $\sum t(i) \le \sum c(i)$ (true after any i) * only upper bound
   - → invariant: always enough credit to pay off any next op
   - → cheap $t(i)$ → overpay $c(i)$
   - → expensive $t(i)$ → lower $c(i)$

   set $c(i)$ as low as possible
   1. tighter bound
   2. analysis easier (credit nearer to 0)
3. Potential Method
   - → cheap op → $\Delta\phi(i) > 0$
   - → expensive op → $\Delta\phi(i) < 0$
   - → $\phi(i)$ is "credit amount" after i-th op
   - * state & check
     1. $\phi(0) = 0$
     2. $\phi(i) \geq 0 \ \forall i$ to be valid ϕ function

$$c(i) = t(i) + \phi(i) - \phi(i-1)$$

amortized cost = actual cost + $(\phi(n) - \phi(0))$

* is an upper bound

Dynamic Programming

→ polynomial subproblems; huge overlap of subproblems

→ iterative bottom-up fashion w/ DP table [easier to prove run-time]

Recursive Formulation: base cases + inductive cases

Optimal Substructure: an optimal soln to a problem contains optimal soln to subproblems

→ prove w/ cut & paste

1. suppose exists a soln that is NOT using "optimal" is optimal
2. add $x[i]$ → even "better"; swap to use "optimal" instead
3. solution is improved / still optimal, contradiction / OK to swap

Overlapping subproblems

Greedy

→ one subproblem at each step → "locally optimal is also globally optimal"

Optimal substructure

→ prove w/ cut & paste

Greedy-choice property: exists an optimal soln that makes the greedy choice

→ prove w/ cut & paste

Max Flow

(1) capacity: $\forall (u,v) \in E: 0 \le f(u,v) \le c(u,v)$

(2) flow conservation: for $u \in V - \{s,t\}$: $\underbrace{\sum f(v,u)}_{\text{flow-in}} = \underbrace{\sum f(u,v)}_{\text{flow-out}}$

Ford - Fulkerson

DFS: $O(|E| \cdot |\text{max flow}|)$; Edmond Karp's: $O(|V||E|^2)$; Dinic's: $O(|V|^2|E|)$

if $(u,v) \in E$: $c_f(u,v) = c(u,v) - f(u,v)$ [can increase by c_f]

if $(v,u) \in E$: $c_f(u,v) = f(v,u)$ [can reduce by c_f]

if neither: $c_f(u,v) = 0$

1. All $f(u,v) = 0$
2. Choose a path from $s \to t$
   1. $m = \min c_f(u,v)$ on path
   2. if $m = 0$, skip path
   3. for each (u,v) in path:
      1. if $(u,v) \in E$, increment $f(u,v)$ by m
      2. if $(v,u) \in E$, decrement $f(v,u)$ by m

→ always integral if inputs are integral

Minimum Cut

$S = \{\text{vertices reachable from } s\}$

$T = V - S$

max flow = $\sum c(u,v)$ for $u \in S, v \in T$

Linear Programming

poly(n, m) if optimal soln exist
→ unbounded
→ infeasible [no possible soln]

→ not necessarily integral soln

Variables: $x_1, \dots, x_n$

Maximise: $\sum_{j \in [n]} c_j \cdot x_j$

$\boxed{\text{Simplex always at vertex}}$

$|x| = y$ → $x \le y$ and $x \ge -y$ and $y \ge 0$

Constraints: $x_1, \dots, x_n \boxed{\geq} 0$ [non-negative]

$a_{11}x_1 + \dots + a_{1n}x_n \boxed{\le} b_1$ } m $\boxed{\text{linear}}$ constraints [no lg, power, xy etc]

min → max, $>, \geq, =$ → $\le$: multiply by -1

not non-neg : replace x w/ $(x' - x'')$

Reductions

if B is easy, A is easy

1. $A \le_p B$ [transitive]
2. A reduces to B in p(n) - time
3. A can be solved in polynomial time using black box for B
   - → prove like typical
4. Can transform instance of A → instance of B
   - → show it's in p(n)
   - → show YES-instance of A ⟷ YES instance of B

Factor, Knapsack DP → pseudo-poly: poly in numeric value & exponential in length

Assigning bits to objects OR choose subsets

→ SAT or Partition

① but small fixed set

→ k Color, 3 Color

Arrange in order

→ Dir / Undir Hamiltonian OR TSP

Small subset satisfying constraints

→ Min Vertex Cover

Large subset satisfying constraints

→ Max Ind Set OR Max Clique OR Max SAT

Partition → Partition

number "3" appears → 3SAT OR 3Color

NP

$P \subseteq (NP \cap co\text{-}NP)$ Factor $\in (NP \cap co\text{-}NP)$

Subset Sum → Partition : add sum(S), 2T

P (deterministic polynomial): can solve in poly time of instance

NP (non-deterministic polynomial): verify YES in poly time of instance

co-NP: verify NO in poly time of instance ← ∃ YES/NO certs ∀ instances → V(instance, certificate)

A is NP-hard if $\forall B \in NP: B \le_p A$ [$NP \le_p$ NP-hard]

NP-complete = NP and NP-hard [equivalent to 3-SAT, show NP + reduce known NP-complete to this]

3-SAT: $(x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_4 \vee x_5) \wedge \cdots$

CNF: $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_n$

Approximation

→ CANNOT reduce in general

C^* : cost of optimal

C : cost found by approx. algo. [worst / expected case]

minimisation: $\frac{C}{C^*} > 1$

maximisation: $\frac{C^*}{C} > 1$

} ideally small & constant

Pick arbitrary ___

let A be set ___ chosen

... so $|A| \le C^*$

since $C = k|A|$, $C \le kC^*$

⇒ k-approximation

Polynomial-Time Approximation Scheme (PTAS): run in poly(n) f(ε) and approx ratio $(1+\varepsilon)$, $\varepsilon > 0$

Knapsack approx → Fully Polynomial-Time Approximation Scheme (FPTAS): run in poly(n, 1/ε) and approx ratio $(1+\varepsilon)$, $\varepsilon > 0$

Diagonal DP

```
for (Δ = 1 to n-1)
    for (i = 1 to n-1-Δ)
        j = i+Δ
```

Approx Proof: ① correct soln ② approx ratio

Greedy interval: start last or ends first

LCS DP

$LCS(i, 0) = \phi$ $LCS(0, j) = \phi$

if $a_n = b_m$, $LCS(n,m) = LCS(n-1, m-1) + a_n$

if $a_n \neq b_m$, $LCS(n,m)$ = bigger of $LCS(n-1, m)$ or $LCS(n, m-1)$

Max Flow LP

vars: $f(u,v)$ for $(u,v) \in E$

maximise $\sum_{v \in \text{neigh}(s)} f(s,v)$

→ $\forall (u,v) \in E: f(u,v) \geq 0$

→ $\forall (u,v) \in E: f(u,v) \le c(u,v)$

→ $\forall u \in V \setminus \{s,t\}: \sum_{v \in V} f(u,v) = \sum_{v \in V} f(v,u)$