Chapter 1 Some Representative Problems

Definition. ***Problems***

1. ***Stable Matching*** *Consider $M = \{m_1, \cdots, m_n\}$ of n men, and a set $W = \{w_1, \cdots, w_n\}$ of n women. Let $M \times W$ denote set of all possible ordered pairs of form (m, w), where $m \in M$ and $w \in W$.*

   (a) ***Matching*** *S is a set of ordered pairs, each from $M \times W$, with property that each member of M and each member of W appears in at most one pair in S.*

   (b) ***Perfect Matching*** *S' is a matching with property that each member of M and each member of W appears in exactly one pair in S'. (Marriage analalogy, everyone is married, no singlehood nor polygamy)*

Chapter 2 Basics of Algorithm Analysis

Definition. ***Concepts***

1. ***Worst-case running time*** *A bound on the largest possible running time the algorithm could have over all inputs of a given size N, and see how this scales with N*

2. ***Average-case Analyiss*** *Performance of algorithm averaged over random instances*

3. ***Efficient Algorithm*** *An algorithm is efficient if it achieves quantitatively better than worst-case performance, at an analytical level than brute-force search. Alternatively, its efficient if it has the algorithm has a polynomial running time*

4. ***Poynomial Running time*** *An algorithm whose running time $T(n) = \mathcal{O}\left(n^d\right)$ for some constant $d \in \mathbb{R}$, independent of input size n*

5. ***Asymptotic Upper Bounds*** *Let $T(n)$ and $f(n)$ be functions, then T is asymptotically upperbounded by f, i.e. $T(n) = \mathcal{O}(f(n))$, if there exists constants fixed $c > 0$ and $n_0 \geq 0$ so that for all $n \geq n_0$, $T(n) \leq cf(n)$*

6. ***Asymptotic Lower Bound*** *$T(n)$ is asymptotically lowerbounded by $\Omega(f(n))$, i.e. $T(n) = \Omega(f(n))$ if there exists constants fixed $\epsilon > 0$ and $n_0 \geq 0$ so that for all $n \geq n_0$, $T(n) \geq \epsilon f(n)$*

7. ***Asymptotic Tight Bounds*** *If $T(n)$ is both $\mathcal{O}(f(n))$ and $\Omega(f(n))$, then $T(n) = \Theta(f(n))$*

8. ***Asymptotics Properties***

   (a) ***Transitivity***

      i. *if $f = \mathcal{O}(g)$ and $g = \mathcal{O}(h)$, then $f = \mathcal{O}(h)$*

ii. *if* $f = \Omega(g)$ *and* $g = \Omega(h)$*, then* $f = \Omega(h)$

iii. *if* $f = \Theta(g)$ *and* $g = \Theta(h)$*, then* $f = \Theta(h)$

(b) ***Sum of Functions***

i. $f = \mathcal{O}(h)$ *and* $g = \mathcal{O}(h)$*, then* $f + g = \mathcal{O}(h)$

ii. *If* $g = \mathcal{O}(f)$*, then* $f+g = \Theta(f)$*.* f *is asymptotically tight bound for combined function* $f + g$

9. ***Asymptotics for Common Functions***

(a) ***Polynomial*** *Let* f *be a polynomial of degree* d*, in which* a_d *is positive. Then* $f = \Theta\left(n^d\right)$

(b) ***Logarithms***

i. *For every* $b > 1$ *and every* $x > 0$*, we have* $\log_b n = \mathcal{O}(n^x)$

ii. $\log_a n = \Theta(\log_b n)$ *(base of logarithm not important)*

(c) ***Exponentials***

i. *For every* $r > 1$ *and every* $d > 0$*, we have* $n^d = \mathcal{O}(r^n)$

ii. *Asymptotically, exponential functions with different base are all differen*

Chapter 7: Network Flow

Definition. ***Concepts***

1. ***Flow Network***

Chapter 13: Randomized Algorithm

13.1 Contention Resolution

Definition. ***Asymptotic of natural number***

$$e^x = \lim_{n\to\infty}\left(1 + \frac{x}{n}\right)^n$$

Definition. ***The Union Bound*** *The probability of a union of event is upper-bounded by the sum of their individual probabilities. Given vents* $\mathcal{E}_1, \mathcal{E}_2, \cdots, \mathcal{E}_n$*, we have*

$$\mathbb{P}\left(\bigcup_{i=1}^{n} \mathcal{E}_i\right) \leq \sum_{i=1}^{n} \mathbb{P}(\mathcal{E}_i)$$

13.2 Finding the Global Minimum Cut

Theorem. *There is a polynomial time algorithm to find a global min-cut in an undirected graph G. The idea is we can formulate the problem of finding global min-cut in undirected graph as finding min s-t cuts in directed graphs $n-1$ times.*

Definition. ***Concepts***

1. ***Multigraphs*** *$G=(V,E)$ is an undirected graph that allows multiple parallel edges between same pair of nodes.*
2. ***Contraction Algorithm*** *Pick uniformly at random an edge $e=(u,v)\in E$ and contract it by producing a new graph G' with given update rules*
   (a) *u,v identified into a single node w*
   (b) *All other nodes keep identity*
   (c) *Edges with one end equal to u and other equal to v deleted from G'*
   (d) *Other edges preserved in G', with end upadted to w, if one if its end is equal to u or v.*

   The algorithm terminates when G' has only two supernodes v_1 and v_2, where we output the cut $(S(v_1),S(v_2))$, where $S(v_1),S(v_2)\subseteq V$ are set of nodes that have been contracted into v

Theorem. ***Lower Bound of Contraction Algorithm is Polynomially Small*** *The Contraction Algorithm returns a global min-cut of G with probability at least $1/\binom{n}{2}=2/n(n-1)$.*

Proof. Proof consists of identifying bad events, i.e. an edge in a min-cut is contracted, and (upper) bounding their probabilities. □

Theorem. *An undirected graph $G=(V,E)$ on n nodes has at most $n2$ global min-cuts*

Definition. ***Number of Global Minimum Cuts*** *An undirected graph $G=(V,E)$ on n nodes has at most $\binom{n}{2}$ global min-cuts*

13.3 Random Variables and Their Expectations

Definition. ***Geometric Series***

$$\sum_{k=0}^{\infty} ar^k = \frac{a}{1-r} \qquad |r|<1$$

Definition. ***Concepts***

1. ***Random Variable*** *X is a function from the underlying sample space to the natural numbers, such that for each natural number j, the set $X^{-1}(j)$ of all sample points taking value j is an event. $\mathbb{P}(X = j)$ is equivalent to writing $\mathbb{P}\left(X^{-1}(j)\right)$.*

2. ***Expectation*** $\mathbb{E}\{X\} = \sum_{j=0}^{\infty} j\mathbb{P}(X = j)$

3. ***Linearity of Expectation*** *Given two random variable X and Y defined over the same probability space, we define $X + Y$ be random variable equal to $X(\omega) + Y(\omega)$ on a sample point ω. For any X and Y, we have*

$$\mathbb{E}\{X + Y\} = \mathbb{E}\{X\} + \mathbb{E}\{Y\}$$

13.6 Hashing: A Randomized Implementation of Dictionaries

Definition. ***Concepts***

1. ***Dictionary Data Structure*** *Given a universe U of possible elements, the data structure try to keep track of $S \subseteq U$ where $|S| << |U|$. The goal is to be albe to insert and delete element from S quickly and determine if a given element belong to S. Note the dictionary is for situations where U is enormous, so storing by index to an array, say, would be infeasible*

2. ***Hash Function*** *Given $S \subseteq U$, where $|S| = n$, we will set up an array H of size n to store the information, and use a function*

$$h : U \to \{0, 1, \cdots, n-1\}$$

that maps elements of U to array positions. h is a hash function and H is a hash table.

3. ***Collision*** *happens when there exists distinct $u, v \in S$, where $h(u) = h(v)$*

4. ***Linked List*** *Each position $H[i]$ of hash table stores a linked list of all $u \in S$, with $h(u) = i$. `Insert` adds u to linked list at position $H[h(u)]$, `Delete` scans the list and removes u if it is present*

5. ***Uniform Random Hashing*** *For each $u \in U$, when inserting u into S, select a value $h(u)$ uniformly at random in the set $\{0, 1, \cdots, n-1\}$, independently of all previous choices. In this case, the probability that two randomly selected values $h(u)$ and $h(v)$ collide is exactly $1/n$*

6. ***Universal CLasses of Hash Functions*** *Choose a function at random from a selected set of functions. Each function $h \in \mathcal{H}$ maps U into the set $\{0, 1, \cdots, n-1\}$, with two additional properties*

(a) ***Universal*** *For any $u, v \in U$, the probability that a randomly chosen $h \in \mathcal{H}$ satisfies $h(u) = h(v)$ is at most $1/n$*

(b) *Each $h \in \mathcal{H}$ can be compactly represented and, for a given $h \in \mathcal{H}$ and $u \in U$, we can compute value $h(u)$ efficiently*

7. ***Gurantees of Universal Classes of Hash Functions*** *Let $\mathcal{H}$ be a universal class of hash functions mapping a universe U to the set $\{0, 1, \cdots, n-1\}$, let S be an arbitrary subset of U of size at most n, and let u be any element in U. We define X to be a random variable equal to the number of elements $s \in S$ for which $h(s) = h(u)$, for a random choice of hash function $h \in \mathcal{H}$ (S and u fixed, randomness is in choice of $h \in \mathcal{H}$). Then*

$$\mathbb{E}\{X\} \leq 1$$

8. ***Running time*** *for* `Lookup`, `Insert`, *and* `Delete` *have running time proportional to time to compute $h(u)$ in addition to the length of linked list at $H[h(u)]$. Previous points imply that these operations have expected running time of $\mathcal{O}(1)$*

13.7 Finding the Closest Pair of Points: Randomized Approach

Definition. ***Concepts***

1. ***Goal*** *Use dictionary, find closest pair of points in $\mathcal{O}(n)$ expected time, plus $\mathcal{O}(n)$ expected dictionary operations. The separation of these two components corresponds to two places where randomization take place, the former in how the algorithm processes input, and the latter in how hashing introduces additional source of randomness as part of hashtable operations*

31.1 Elementary number-theoretic notions

32.2 The Rabin-Karp Algorithm

Definition. ***Concepts***

1. ***String Matching*** *Given text $T[1..n]$ and pattern $P[1..m]$ where $m \leq n$. Assume elements of P and T are drawn from alphabet Σ. P occurs with **shift** s in text T if $T[s+j] = P[j]$ for all $1 \leq j \leq m$, i.e. $P[1..m] = T[s+1..s+m]$. If P occurs with shift s in T, s is a **valid shift**. The string-matching problem deals with finding all valid shift with a gaiven pattern P occurs in a given text T*

2. ***Naive Algorithm*** *Checks for condition $P[1..m] = T[s+1..s+m]$ for all $s = 0, \cdots, n-m$, a total of $n - m + 1$ possible values of s.*

3. ***Rabin-Karp Algorithm***

   (a) *$\Theta(m)$ preprocessing, $\Theta((n-m+1))$ matching worst case.*

(b) ***When*** m ***is small*** $\Theta(m)$ *preprocessing time,* $\Theta((n-m+1))$ *matching time. Use* ***Horner's rule*** *to convert* P *to decimal representation* p *and each* $t_0, \cdots, t_{n-m}$*, where* t_s *is decimal representation for* $T[s+1..s+m]$

i. *Computing* p *takes* $\Theta(m)$

ii. *Computing* t_0 *takes* $\Theta(m)$

iii. *Computing* t_{s+1} *from* t_s *takes* $O(1)$*, so together takes* $\Theta(n-m)$

iv. *Comparing* p *with each of* t_s *takes* $\Theta(n-m)$

(c) ***When*** m ***is large****, arithmetic operation on* p *not constant.*

i. $\Theta(m)$ *preprocessing,* $\Theta((n-m+1)m)$ *matching worst case.*

ii. *All characters interpreted as radix-d digits*

iii. ***Heuristic to rule out invalid shift*** *Compute* $p \mod q$ *and* $t_s \mod q$*. If* $p \mod q \neq t_s \mod q$*, then* $p \neq t_s$*, so* s *is invalid. Computing modulo does not change time complexity.*

iv. ***Test for Spurious Hit*** *However if* $t_s \mod q = p \mod q$ *does not imply* $t_s = p$*, so need to test further to see whether* s *is really valid or a spurious hit. Idea is this occurs infrequently*

Definition. ***Binomial Series***

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}$$