Randomized Algorithms
Week 4: Decision Problems

Rao Kosaraju

4.1 Decision Problems

Definition 1. **Decision Problem:** For a language $L$ over an alphabet $\sum$, given any $x \in \sum^*$, is $x \in L$.

Definition 2. **Las Vegas Algorithm:** Algorithm $A$ for language $L$ is a Las Vegas algorithm if for every input $x \in \sum^*$, $A$ correctly outputs whether $x \in L$ or not.

In designing such algorithms, the interest is in deriving expected speed and with high probability results.

Definition 3. **Monte Carlo Algorithm:** Algorithm $A$ for language $L$ is a Monte Carlo algorithm if for any $x \in \sum^*$, $A$ is permitted to make mistakes with some probability. Two important classes of Monte Carlo algorithm are defined below:

a) **RP Algorithm (Randomized Polynomial Time):**

For every $x \in \sum^*$,

- $x \notin L \Rightarrow$ On input $x$, algorithm $A$ outputs no, and
- $x \in L \Rightarrow$ On input $x$, algorithm $A$ outputs yes with probability $\geq \frac{1}{2}$.

The function $\frac{1}{2}$ is not critical, it can be any constant in the range of $(0,1]$.

Thus a RP algorithm can output no when the correct answer is yes. Whenever it outputs ‘yes’, the answer is correct; i.e. $x \in L$.

a.1) **Co-RP Algorithm:** is an RP algorithm for $\overline{L}$.

RP and co-RP algorithms are examples that have one-sided error.
b) BPP Algorithm (Bounded-error Probabilistic Polynomial time):

There exists an \( \epsilon > 0 \) such that

\[ x \in L \Rightarrow \text{On input } x, \text{ algorithm } A \text{ outputs yes with probability } \geq \frac{1}{2} + \epsilon. \]

\[ x \in \bar{L} \Rightarrow \text{On input } x, \text{ algorithm } A \text{ outputs no with probability } \geq \frac{1}{2} + \epsilon. \]

Thus on every \( x \in \Sigma^* \), algorithm \( A \) can output the incorrect answer with probability \( < \frac{1}{2} - \epsilon \). RP algorithms can make one-sided errors, while BPP algorithms make two-sided errors.

4.2 Computation Problems

In a computation problem, the required output is a set of values instead of a simple yes or no response. There are 2 broad classes of algorithms for computation problems.

a) Algorithms that compute exact output. RandQuickSort is an example of this class of algorithms.

b) Algorithms that compute approximate output. Algorithms for set balancing, cyclic shift, Hamming center and hitting set are algorithms of this class. Let the optimum value for the input be \( \text{opt} \). If it is a minimization problem we seek an algorithm that achieves a value \( \alpha \text{opt} \) in which \( \alpha \geq 1 \) and is as small as possible. For maximization problems the algorithm needs to achieve \( \alpha \text{opt} \) in which \( \alpha \leq 1 \) and is as large as possible.

Existence Arguments:

Our main goal is algorithms for solving problems. Some problems deal with constructing graphs or functions with certain properties. If we fail to construct randomized algorithms, the next best thing we can do is to establish that graphs or functions with the desired properties do exist. We make use of simple probabilistic arguments in establishing such an existence. Two such principles are:

**Observation 1.** For any event \( E \), if \( P(E) > 0 \) then \( E \) is non-empty; i.e. \( E \) contains at least one element.

**Observation 2.** For any random variable \( X \) with expectation \( \mu_X \), \( X \) must take a value \( \geq \mu_X \) and also take a value \( \leq \mu_X \).

We illustrate these principles with several examples.

**Definition 4. Max Cut Problem:** Given an undirected graph, partition the vertices into 2 blocks such that the number of edges that span the blocks is maximized.

By making use of Observation 2, we show that in any undirected graph there exists a cut...
of size $\geq \frac{m^2}{2}$, where $m$ is the number of edges of the graph.

We partition the vertices into 2 blocks, blocks 0 and 1, as follows:
Choose 0, 1 valued r.v.s $X_1, X_2, \ldots, X_n$ independently and uniformly at random. For each $i$, assign the $i^{th}$ vertex to block 0 of $X_i = 0$ and to block 1 if $X_i = 1$.

For each edge $e = (i, j)$, let
$$Y_e = \begin{cases} 
1 & \text{if } i \text{ and } j \text{ are assigned to different blocks} \\
0 & \text{otherwise}
\end{cases}$$

Then $Y = \sum_e Y_e$ counts the size of the cut.

Note that $P(Y_e = 1) = \frac{1}{2}$ since $X_j$ gets the same value as $X_i$ with probability $\frac{1}{2}$.

Hence, $E(Y_e) = \frac{1}{2}$. Hence, $E(Y) = \frac{m^2}{2}$.

Hence, by Observation 2, $Y$ can take a value $\geq \frac{m^2}{2}$. Hence there exists a cut of size $\geq \frac{m^2}{2}$.

In fact, it is not too hard to design even a deterministic algorithm that realizes a cut of size $\geq \frac{m^2}{2}$. So this problem doesn’t fully illustrate the power of the probabilistic argument. The next problem is a better illustration of the strength of the probabilistic argument.

**Definition 5.** In a graph $G$, for any set $S$ of vertices let $\Gamma(S) = \{ j \mid j \notin S \text{ and there exists an } i \in S \text{ such that } (i, j) \in E \}$. An $(n, d, c)$ bipartite expander is an $n$ times $n$ bipartite multi-graph $(L, R)$ such that

1) every vertex has degree $d$,
2) every $S \subseteq L$ with $|S| \leq \frac{n}{2}$, satisfies $|\Gamma(S)| \geq c|S|$, and
3) every $S \subseteq R$ with $|S| \leq \frac{n}{2}$, satisfies $|\Gamma(S)| \geq c|S|$.

**Theorem 1.** There exist an $(n, 14, 2)$ bipartite expander.

**Proof.** Conduct the following probabilistic experiment:

1) Choose a random permutation $(i_1, i_2, \ldots, i_n)$ of $\{1, 2, \ldots, n\}$ u.a.r. Then for every $j$,
Connect vertex $j$ of $L$ to vertex $i_j$ of $R$

2) Repeat the above independently $d = 14$ times.

(A way of choosing a random permutation with u.a.r. is the following. In the first step, for vertex 1 of $L$ choose a vertex in $R$ uniformly at random. At the $i^{th}$ step, for vertex of $L$
choose a vertex in \( R \) uniformly at random from the vertices not yet chosen.)

Now we categorize the event that the resulting graph is not an \((n, 14, 2)\) bipartite expander as the union of several events.

\[
a) (\exists s \leq \frac{n}{2c}) (\exists S \subseteq L \text{ of size } s) (\exists S' \subseteq R \text{ of size } cs - 1) (\Gamma(S) \subseteq S'),
\]

\[
b) (\exists s \leq \frac{n}{2c}) (\exists S \subseteq R \text{ of size } s) (\exists S' \subseteq L \text{ of size } cs - 1) (\Gamma(S) \subseteq S').
\]

For computational simplicity, in the following, we make the size of \( R \) to be \( cs \). This only makes the failure probability to go up.

We fix \( s \leq \frac{n}{2c} \), set \( S \subseteq L \), such that \(|S| = s\), and a set \( S' \subseteq R \) such that \(|S'| = cs\) and for that \( S \) and \( S' \) we compute the probability that all edges from \( S \) fall into \( S' \). We then invoke Boole’s inequality and add the probabilities of all such sets \( S \) and \( S' \) for all possible values of \( s \). We repeat this with the roles of \( L \) and \( R \) reversed.

\[
P(\text{all the edges out of } S \text{ fall within } S') = \left(\frac{cs(cs - 1)....(cs - s + 1)}{n(n-1)....(n-s+1)}\right)^d
\]

\[
\leq \left(\frac{cs}{n}\right)^{sd}
\]

There are \( \binom{n}{s} \) ways of choosing \( S \) and \( \binom{n}{cs} \) ways of choosing \( S' \).

\[
P(\text{for some } S \subseteq L \text{ of size } s \text{ and } S' \subseteq R \text{ of size } cs \text{ all edges out of } S \text{ fall within } S') \leq \binom{n}{s} \binom{n}{cs} \left(\frac{cs}{n}\right)^{sd}
\]

\[
\leq \left(\frac{ne}{s}\right)^s \left(\frac{ne}{cs}\right)^{cs} \left(\frac{cs}{n}\right)^{sd}
\]

\[
= \left[\left(\frac{cs}{n}\right)^{d-c-1} c^{1+e}]^s
\]

\[
\leq \left[\frac{1}{2}\right]^{d-c-1} c^{e+1} c^s
\]

Hence probability of left to right failure \( \leq \sum_{s=1}^{n} \left[\left(\frac{1}{2}\right)^{11} c^3 2\right]^s. \)
By symmetry, probability of failure when $S \subseteq R$ and $S' \subseteq L$ has the same value. Hence $P(\text{the graph fails to be an } (n, 14, 2)\text{ bipartite expander}) \leq 2\sum_{s=1}^{n/2^r} |(\frac{1}{2})^{11} e^3 2^s| < 1.$

Hence with positive probability the graph is an $(n, 14, 2)$ bipartite expander. Hence there exists an $(n, 14, 2)$ bipartite expander.

\[\square\]

### 4.3 Number of Hash Functions

**Definition 6.** Let a universe $U$ be of size $m$. Let $S \subseteq U$ be size $n$ and $R$ be an array of $n$ locations. For any function $f : U \rightarrow R$ and $1 \leq i \leq n$, let $n_i$ elements of $S$ map into location $i$ of $R$. The function $f$ is said to be acceptable for $S$, if $\sum_{i=1}^{n} n_i^2 \leq 2(2n - 1)$.

**Theorem 2.** There exist $d = n \log_2 m + 1$ functions $g_1, g_2, \ldots, g_d$ s.t. $(\forall S \subseteq U) (\exists g_j) (g_j$ is acceptable for $S)$.

**Proof.** Choose $d$ random functions $g_1, g_2, \ldots, g_d : U \rightarrow R$ independently and uniformly at random. Fix an $S \subseteq U$ of size $n$. For any $i$, let the function $g_i$ map, for any $1 \leq j \leq n$, $X_{ij}$ elements of $S$ into location $j$ of $R$.

Let $X_i = \sum_{j=0}^{n-1} X_{ij}^2$

We have proved earlier that $E(X_i) = E(\sum_{j=1}^{n} X_{ij}^2) = 2n - 1$.

Our goal is to show that, there exist $d$ functions $g_1, g_2, \ldots, g_d$ such that

$$(\forall S \subseteq U \text{ of size } n) (\exists i) (g_i \text{ is acceptable for } S).$$

Hence failure happens if $(\exists S \subseteq U \text{ of size } n) (\forall i) (g_i \text{ is not acceptable for } S)$.

$P(\text{every } g_i \text{ is not acceptable for } S) = P(\text{results in } \sum_{j=1}^{n} X_{ij}^2 > 2(2n - 1)) \leq \frac{1}{2}$ (by Markov inequality)

$$P(\text{every } g_i \text{ is not acceptable for } S) \leq \frac{1}{2^d}.$$  

Note that there are $\binom{m}{n}$ ways of choosing the set $S$. Hence,

$$P((\exists S) \text{ (every } g_i \text{ is not acceptable for } S)) \leq \binom{m}{n} \frac{1}{2^d} \text{ (by Boole's inequality).}$$

$$\leq m^n \frac{1}{2^d} < 1.$$  

5
Hence there exist $g_1, \ldots, g_d$ such that for every $S$ there exists an acceptable function $g_i$ acceptable for $S$. 