Probabilistic Algorithms.

In this algorithms, we have some use of probability. Either we can say that the algorithm computes a solution always correctly with an expected time which is small (polynomial), or it computes the correct solution with some probability in polynomial time. This gives us the two paradigms:

Monte Carlo Algorithm: An algorithm A which terminates in polynomial time and outputs a correct solution with a probability $p > 0$ on each instance.

Las Vegas Algorithm: An algorithm A which always outputs correct solution and runs in polynomial expected time.

Fact. If there is a Monte Carlo algorithm for a problem P with probability of correct solutions being $p > 0$, and if the solutions can be verified in polynomial time, then there is a Las Vegas algorithm for problem P.

Proof. - We run the Monte Carlo algorithm M multiple times till we get a correct solution.

- The expected number of times to run M is: $\sum_{i=1}^{\infty} i p (1-p)^{i-1}$

Claim. $\sum_{i=1}^{\infty} i p (1-p)^{i-1} = \frac{1}{p}$

Proof. $\sum_{i=0}^{\infty} i p (1-p)^{i-1} = p \sum_{i=0}^{\infty} i (1-p)^{i-1}$

$= p \sum_{i=0}^{\infty} -\frac{d}{dp}(1-p)^i$

$$\sum_{i=0}^{\infty} i\,p(1-p)^{i-1} = p\sum_{i=0}^{\infty} -\frac{d}{dp}(1-p)^{i}$$

$$= -p\frac{d}{dp}\left(\sum_{i=0}^{\infty}(1-p)^{i}\right)$$

$$= -p\frac{d}{dp}\left(\frac{1}{1-(1-p)}\right) \quad \text{since } (1-p) < 1$$

$$= +p \cdot \frac{1}{p^2}$$

$$= \frac{1}{p}$$

- therefore, the expected running time of the algorithm is $\leq \frac{1}{p}$ (time to run each round + time to verify solution)

- assuming p to be a constant, we have expected running time which is polynomial.

It is also true that if we have a Las Vegas algorithm for a problem, then we can have a Monte Carlo algorithm.

Fact 2. If M is a Las Vegas algorithm for a problem P with expected time $E(n)$, then there is a Monte Carlo algorithm for P with probability

Proof. Let t be the maximum running time for which we run M and accept the solution if it terminates before the deadline.

Then, $\text{Prob.}(t > p \cdot E(n)) \leq \frac{1}{p} \cdot \frac{E(n)}{E(n)}$

$= \frac{1}{p}$ for some $p > 1$.

We have now a Monte Carlo algorithm with Prob. of success $\leq \frac{1}{p}$, which run in $O(E(n))$ time.

Fact 2. If M is a Las Vegas algorithm for a problem P with expected time $E(n)$, then there is a Monte Carlo algorithm for P with probability

Proof. Let t be the maximum running time for which we run M and accept the solution if it terminates before the deadline.

Then, $\text{Prob.}(t > p \cdot E(n)) \leq \frac{1}{p} \cdot \frac{E(n)}{E(n)}$

$= \frac{1}{p}$ for some $p > 1$.

We have now a Monte Carlo algorithm with Prob. of success $\leq \frac{1}{p}$, which run in $O(E(n))$ time.

Max Cut Approximation

We study a probabilistic algorithm for a well known NP-hard problem called Max Cut.

Max Cut: Given a graph $G=(V,E)$, partition V into two sets, A and B, so that the # of edges crossing the cut $V=A \dot{\cup} B$ is maximized.

First, we show that the maximum cut has at least $m/2$ edges where $m=|E|$.

Theorem. Let $C(A,B)$ be the size of cut for $V=A \dot{\cup} B$. Then, $\max_{A,B} C(A,B) \geq \frac{m}{2}$.

Proof.

- Construct A, B randomly and independently assigning each vertex to one of A & B.
- Let $e_1, \ldots, e_m$ be the edges in G.
- Define X_i a random variable $X_i = \begin{cases} 1 & \text{if } e_i \text{ is in cut} \\ 0 & \text{otherwise} \end{cases}$

- $E[x_i] = \frac{1}{2}$ because the prob. that endpoints of e_i are in different sets is $\frac{1}{2}$.

- Let $C(A,B)$ be the random variable denoting the value of the cut $V = A \dot{\cup} B$.

- $$E[C(A,B)] = E\left[\sum_{i=1}^{m} x_i\right]$$
$$= \sum_{i=1}^{m} E[x_i]$$
$$= m \cdot \frac{1}{2} = \frac{m}{2}.$$

- Since the expection of the cut size is $\frac{m}{2}$, there exists a cut with size $\geq \frac{m}{2}$.

The above proof also provides a Las Vegas algorithm for finding a cut of size at least $\frac{m}{2}$.

- $E[x_i] = \frac{1}{2}$ because the prob. that endpoints of e_i are in different sets is $\frac{1}{2}$.

- Let $C(A,B)$ be the random variable denoting the value of the cut $V = A \dot{\cup} B$.

- $$E[C(A,B)] = E\left[\sum_{i=1}^{m} x_i\right]$$
$$= \sum_{i=1}^{m} E[x_i]$$
$$= m \cdot \frac{1}{2} = \frac{m}{2}.$$

- Since the expection of the cut size is $\frac{m}{2}$, there exists a cut with size $\geq \frac{m}{2}$.

The above proof also provides a Las Vegas algorithm for finding a cut of size at least $\frac{m}{2}$.

Algorithm.

Simply flip a coin and put a vertex with probability $\frac{1}{2}$ into A or B.

The above algorithm runs in polynomial time, and produces a cut whose expected size $\frac{m}{2}$.

Let $p = \text{Prob.}\left(c(A,B) \geq \frac{m}{2}\right)$ and observe

$$c(A,B) \leq m.$$

$$\frac{m}{2} = E[c(A,B)]$$

$$= \sum_{i \leq m/2 - 1} i \,\text{Prob.}(c(A,B) = i) + \sum_{i \geq m/2} i \,\text{Prob.}(c(A,B) = i)$$

$$\leq (1-p)\left(\frac{m}{2} - 1\right) + pm$$

$$\Rightarrow p \geq \frac{1}{\frac{m}{2} + 1}.$$

Therefore, the probability that the algorithm produces a cut of size at least $\frac{m}{2}$ is more than $\frac{1}{\frac{m}{2}+1}$.

- This is a Monte Carlo algorithm.

- We can get the Las Vegas algorithm using the trick we discussed before

- This tells us that we have a Las Vegas algorithm ensuring at least $\frac{m}{2}$-size cut in time whose expection is polynomial...

- Since any cut cannot have size more than m, we have a $\frac{1}{2}$-approximation algorithm.