UM-SJTU JOINT INSTITUTE

Introduction to Algorithms
(VE477)

Homework #6

Prof. Manuel

Xinmiao Yu
518021910792

Nov. 6, 2020

Q1.

1. Denote that $L = l_1, l_2, ..., l_n$ and $R = r_1, r_2, ..., r_n$, $i, j \leq n$. As the matrix A is defined as

$$a_{i,j} = \begin{cases} X_{i,j}, & (l_i, r_j) \in E, \\ 0, & (l_i, r_j) \notin E \end{cases}$$

and the expression of determinant is

$$Det(A) = \sum_{\pi \in S_n} (-1)^{sgn(\pi)} \Pi_{i=1}^{n} a_{i,\pi(i)},$$

where S_n is the set of all permutations on $[n]$ and $sgn(\pi)$ is the sign of the permutation π.

There is a one to one correspondence between a permutation $\pi \in S$ and a possibly exists perfect matching $\{(l_1, r_{\pi(1)}), (l_2, r_{\pi(2)}), ..., (l_n, r_{\pi(n)})\}$. As for not perfect matching case , $\Pi_{i=1}^{n} a_{i,\pi(i)}$ will be zero. Then we could denote the set of perfect matchings in G as P, and the determinant could be rewritten as

$$Det(A) = \sum_{\pi \in P} (-1)^{sgn(\pi)} \Pi_{i=1}^{n} X_{i,\pi(i)}$$

If G **has no perfect matching**, then the set $P = \emptyset$. Therefore, the determinant is identically zero.

If the determinant is identically zero, this only happens when the summation of all the term is zero, which is same as $P = \emptyset$ and no perfect matching exist.

So $Det(A)$ is identically zero if and only if G as no perfect matching.

2. Then to detect whether a perfect matching exist is same as if a multivariate polynomial of degree at most n is equivalent to 0. At most $n!$ terms will be in the $Det(A)$.

Assign random weight for each edge $\in E$ such that$w_{ij} \in \{1, 2, ..., 2m\}$, where $m = |E|$, by the isolating lemma, the minimum weight perfect matching in G will be unique with probability at least 1/2. Set each $X_{i,j} = 2^{w_{ij}}$. Let A_{ij} be the matrix obtained from A by removing the i^{th} row and j^{th} column.

Suppose there is a perfect matching P has a unique minimum weight W. Then with two lemma

- $Det(A) \neq 0$ ans the highest power of 2 that divides $Det(A)$ is 2^W.
- Edge (i, j) belongs to P if and only if $Det(A_{ij}/2^{W-w_{ij}})$ is odd.

Problem 1 continued on next page. . .

Algorithm 1: randomized algorithm to find a perfect matching

Input : graph $G = (V, E)$, $V = L \cup R, L \cap R = \emptyset$

Output: whether exists a perfect matching

Choose n^2 integers from $w_{ij} \in \{1, 2, ..., 2m\}$ randomly
Compute $Det(A)$ by Gaussian elimination
Compute $adj(A)$ and recover each $Det(B_{ij})$
for *each edge* $\in E$ **do**
  if $Det(A_{ij})/2^{W-w_{ij}}$ *is odd* **then**
    add the edge to M
  end if
end for

3. Time complexity $\mathcal{O}(nlogn)$. If we run the algorithm for k times and output $noperfectmatching$ if and only if all says no, then the error probability is 2^{-k}.

4. *The deterministic polynomial time algorithm* is to reduce this problem to the max-flow problem as discussed in the lecture. It is not useful for parallel algorithms.

   Then this algorithm still viewed as useful, as the error probability could be reduced to a rather low level.

$Reference$: $web.stanford.edu/class/msande319/MatchingSpring19/lecture01.pdf$

Q2.

The basic idea is to hold two pointers, one $fast$ and one *slow*.

1. Find the middle one, when even number n of nodes, will return $(n/2 + 1)$-th node (assume the first one as index 1).

Algorithm 2: Find the middle node

Input : the *head* of a singly linked list

Output: the middle node of the list

slow $\leftarrow$ *head*
$fast$ $\leftarrow$ *head*
while $fast$ *is not None* **do**
  $fast \leftarrow fast.next$
  if $fast$ *is None* **then**
    return *slow*
  end if
  $fast \leftarrow fast.next$
  $slow \leftarrow slow.next$
end while
return *slow*

2. Detect whether a list contains a cycle.

Algorithm 3: Detect cycle

Input : the *head* of a singly linked list

Output: whether exists a cycle

```
slow ← head
fast ← head
while fast and slow not None do
    if fast.next is None then
        return False
    end if
    fast ← fast.next.next
    slow ← slow.next
    if fast == slow then
        return True
    end if
end while
return False
```

With the linked list of n nodes, the time complexity is $\mathcal{O}(n)$.

Q3.

1. At least n boxes, for each time obtain a coupon not appeared before.

2. From the definition of X and X_j, $X = X_1 + ... + X_j + ... + X_n$. The probability of collecting coupon j given that already obtain $j-1$ coupon is $P_j = \dfrac{n-(j-1)}{n}$. Therefore, X_j is a geometric distribution with expectation

$$E[X_j] = \frac{1}{P_j} = \frac{n}{n-j+1}$$

3. To prove $E[X] = \Theta(nlogn)$

$$\begin{aligned} E[X] &= E[X_1] + E[X_1] + ... + E[X_j] + ... + E[X_n] \\ &= \frac{n}{n} + \frac{n}{n-1} + ... + \frac{n}{n-j+1} + ... + \frac{n}{1} \\ &= n\sum_{i=1}^{n} \frac{1}{i} \approx n\int_1^n \frac{1}{x}dx \\ &= nlogn \end{aligned}$$

4. Find that for any $1 \le i < j \le n$, $E[X_j] > E[X_i]$. So more coupons have already get, more coupons need to buy for collecting a new kind of coupon.