CS6033 Assign No.6

Minghe Yang

April 2021

1 The partition problem

1.1 The Leibniz formula for the determinant of an $n * n$ matrix A is decribed as follow:

$$det(A) = \sum_{\sigma \in S_n} (sgn(\sigma) \prod_{i=1}^{n} a_{i\sigma(i)}) \tag{1}$$

Where $\prod$ is the set of all permutations of {1,..., n}.The sign$sgn(\sigma)$ of a permutation σ is +1 for even and -1 for odd inverted pairs.
For this matrix A, the value of $\prod_{i=1}^{n} a_{i\sigma(i)}$ will be non-zero if and only if all terms are non-zero. Since the determinant is the sum of these terms, it follows that if det(A) is nonzero there must exist at least one perfect matching in G. But this leads to another situation: Even and odd situations(perfect matching) are same, so it'll still be zero with several perfect matching available.
Luckily in this question $a_{i,j}$ is changed to $X_{i,j}$ and are no longer bounded to 1 or 0. If $X_{i,j}$ is random, if determinant of A is identically zero, we can confirm that each even or odd situations have one element equals to 0. So then we can confirm there is no perfect matching.
1.2

Algorithm 1 Randomized algorithm for perfect matching

Input: Determinant A
Output: Perfect matching available or not
1: set x_{ij} to be a number chosen uniformly at random from $\{1, ..., n^2 m\}$
2: compute $det(B)$
3: if $det(B) = 0$ repeat until confidence is above the desired threshold
4: else return perfect matching available

1.3 The complexity of this algorithm is $O(kn^3)$(k for selected cases and n^3 for each determinant solving), the probility will be $Pr \leq \frac{d}{|S|} = \frac{d}{ld} = \frac{1}{l}$
1.4 The benefit for Randomized Algorithm is that it can reduce time complexity, while it probably can't give you an exact answer(definitely no or probably yes). Since deterministic polynomial time algorithms already exist, if you want an exact answer, better not to choose randomized algorithm.

2 Critical thinking

2.1

Algorithm 2 Middle node in one pass

Input: Singly linked list LL
Output: Middle node slow

```
function M(LL, k)
    slow = LL.head
    fast = LL.head
    while fast and fast.next do
        slow = slow.next
        fast = fast.next.next
    end while
return slow
end function
```

2.2 The complexity of this algorithm is $O(n)$.If there are n nodes, the slow

Algorithm 3 Cycle detection

Input: Singly linked list LL
Output: Whether this list contains a cycle or not

```
function Floyd(LL, x_0)
    tortoise = LL.head
    hare = LL.head
    while tortoise != hare and hare != end of the list do
        tortoise = tortoise.next
        hare = hare.next.next
    end while
    if hare != end of the list then return There is no loop
    elsereturn There's a loop
    end if
end function
```

pointer needs to travel within n steps before the fast pointer either meets the slow pointer or finds the end. That means you do O(n) work.

3 The coupon collector desillusion

3.1 At least n boxes are needed.

3.3 $p^i = \frac{n-i+1}{n}$, so $E(T) = E(t_1 + t_2 + t_3 + t_4 + \cdots + t_n)$
$= \frac{1}{p_1} + \frac{1}{p_2} + \frac{1}{p_3} + \cdots + \frac{1}{p_n}$
$= \frac{n}{n} + \frac{n}{n-1} + \frac{n}{n-2} + \frac{n}{n-3} + \cdots + \frac{n}{1}$
$= n \cdot (\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n})$

$= n \cdot H_n$. Using the asymptotics of the harmonic numbers, we can get:
$E(t) = n \cdot H_n = n \log n + \gamma n + \frac{1}{2} + O(\frac{1}{n})$ So $E[t] = \Theta(nlogn)$

3.4 For example, first day you'll get one exactly different coupon, for the second day, different coupon chance will be $(n-1)/n$, after you collected 2nd coupon, the third chance will be $(n-2)/n$..., for the last coupon, the chance will be $1/n$. t_i is time to collect the i-th coupon, and the probility of a new coupon will be $p_i = \frac{n-i+1}{n}$, Therefore, t_i has geometric distribution with expectation:
$\frac{1}{pi} = \frac{n}{n-i+1}$. So we have expectation of $E(T)$ from above. Then using the asymptotics of the harmonic numbers, we can get
$E(t) = n \cdot H_n = n \log n + \gamma n + \frac{1}{2} + O(\frac{1}{n})$.
Since $n \log n$ is the highest degree, we can conclude that $E[t] = \Theta(nlogn)$