Soham Chatterjee
Email: sohamc@cmi.ac.in
Course: Parallel Algorithms and Complexity

Assignment - 7
Roll: BMC202175
Date: October 3, 2023

Problem 1

Analysis Of Luby's Algorithm

Solution:

Algorithm 1: Luby's Randomized Algorithm on MIS

```
1  begin
2      A ⟵ ∅
3      while G ≠ ∅ do
4          S ⟵ ∅
5          I ⟵ ∅
6          for v ∈ V(G) in parallel do
7              add v to S with probability 1/(2d(v))
8          for {u, v} ∈ E(G) in parallel do
9              if u ∈ S and v ∈ S then
10                 if d(u) < d(v) then
11                     delete u from S
12                 else if d(v) < d(u) then
13                     delete v from S
14                 else if u < v then
15                     delete u from S
16                 else
17                     delete v from S
18         Call the resulting set after deletions I
19         A ⟵ A ∪ I
20         G ⟵ G \ (I ∪ N(I))
21     return A
```

1 Analysis of Randomized Algorithm:

The while loop in line 3 will be executed at most $9\log(n)$ many time since in each iteration we get that the number of edges is expected to get reduced to $\frac{71}{72}$ of edges at the start of iteration. So the outer loop only runs for $\log(n)$ many iterations. The for loop on line 5 takes $O(1)$ parallel time and $O(n)$ processors, as we can use different processor for each vertex simultaneously. The for loop on line 8 takes $O(1)$ time and $O(n^2)$ many processors in $CRCW - PRAM$ model. Since $CRCW(O(1), poly(n)) \subseteq AC^0 \subseteq NC^1$ so it takes parallel depth of $O(\log n)$ to implement this loop. So overall algorithm can be implemented in RNC^2.

2 Analysis of Derandomized Algorithm:

We don't need the vertices to be independent. Pairwise independent is enough for the analysis. We construct pairwise independent family of functions by taking a prime p in the range n to $2n$ (such prime exists because of Bartrand's Postulate) and then we can assume the vertices of the graph are the elements of the finite field $\mathbb{Z}_p$. Now for each vertex u we take $a(u)$ be an integer in $\mathbb{Z}_p$ such that $\frac{1}{2d(u)} \approx \frac{a(u)}{p}$ that

is $\frac{a(u)}{p}$ is as close as possible to $\frac{1}{2d(u)}$. Then we denote $A_u := \{0, 1 \ldots, a(u) - 1\}$. We can take A_u as any subset of $\mathbb{Z}_p$ of size $a(u)$. Hence probability of landing in this set is $\frac{a(u)}{p} \approx \frac{1}{2d(u)}$.

Now we choose x and y uniformly at random $\mathbb{Z}_p$ and define the function

$$f_{x,y} : v \mapsto x + vy \pmod{p}$$

Now for any $u, v, \alpha, \beta \in \mathbb{Z}_p$ where $u \neq v$ then there exists exactly one solution to the linear system

$$x + uy = \alpha \qquad x + vy = \beta$$

since the matrix $\begin{bmatrix} 1 & u \\ 1 & v \end{bmatrix}$ is invertible and

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & u \\ 1 & v \end{bmatrix}^{-1} \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

Therefore

$$\begin{aligned}
\Pr_{x,y \in \mathbb{Z}_p} \left[f_{x,y}(u) \in A_u \wedge f_{x,y}(v) \in A_v \right] &= \Pr_{\substack{\alpha \in A_u \\ \beta \in A_v}} \left[x + uy = \alpha \wedge x + vy = \beta \right] \\
&= \frac{1}{p^2} \left| \{(x, y) \mid x + uy \in A_u \wedge x + vy \in A_v\} \right| \\
&= \frac{1}{p^2} \sum_{\alpha \in A_u} \sum_{\beta \in A_v} \left| \{(x, y) \mid x + uy = \alpha \wedge x + vy = \beta\} \right| \\
&= \frac{1}{p^2} \sum_{\alpha \in A_u} \sum_{\beta \in A_v} 1 \\
&= \frac{1}{p^2} a(u) a(v) = \frac{a(u)}{p} \frac{a(v)}{p} \approx \frac{1}{2d(u)} \frac{1}{2d(v)}
\end{aligned}$$

So we take the family of pairwise independent functions $\mathcal{H} = \{f_{x,y} \mid x, y \in \mathbb{Z}_p\}$. We can construct this family with only $2 \log p = O(\log n)$ random bits (while choosing x, y). Hence there are total $2^{O(\log n)} = n^{O(1)}$ many functions. So $|\mathcal{H}| = n^{O(1)}$.

So we in parallel consider all possible strings of $O(\log n)$ length representing all possible outcomes of the $O(\log n)$ random bits. From each of this string we construct the function in $\mathcal{H}$ and carry on the algorithm deterministically which takes constant depth. Since we expect to delete at least a constant fraction of the edges one of the functions must delete at least that many edges. So we pick the function which deletes the most edges and throw the other parallel computation away and then repeat the whole process. Everything is deterministic and at least a constant fraction of the edges are removed at each stage.

Since constructing the pairwise independent functions takes constant depth and then we can do our algorithm as before for each function in parallel and then take the only necessary function in consideration. So we can get that the derandomized algorithm is in NC^2.

□