Hands-On 6: Algorithm Design A.Y. 2024

F. Boni G. Braccini S. Ghosh G. Trapani

29 April 2024

1 Exercise 1

Show how to use matrix computation to compute the diameter D exactly (time is $O(n^\omega \log D)$, where n^ω is the cost of binary matrix multiplication, where ω is unknown and currently is only known that $2 \leq \omega \leq 2.3728596$).

Call $d(i, j)$ the distance between vertices i and j.

Since we are using adjacency matrices, we can make use of the following property.

Remark. *Let A be the adjacency matrix for a graph G with an added self-loop for each vertex.*
The element (i, j) in the matrix A^n gives the number of walks of length n from vertex i to vertex j; in other terms, the element $A^n[i, j]$ is different from 0 if and only if $d(i, j) \leq n$.

Calling A the adjacency matrix for the graph G with an added self-loop for each vertex, we can now give the pseudocode of the algorithm we can use to compute the diameter:

```
def Diameter(A):
        M_0 ← A;
        d ← 0;
        while ∃(i,j). M_k[i,j] = 0: #we know k ≤ log D from the remark
                M_{k+1} ← M_k^2; #multiplication step takes n^ω
        B ← M_{k-1};
        d ← 2^{k-1};
        #Now we need to binary search for the result
        for i in [k-2; 0]:
                C ← B × M_i;
                if ∃(r,c). M_i[r,c] = 0: #we add this term
                        B ← C;
                        d ← d + 2^i;
        return d;
```

Exercise 2

Prove that any BFS gives a 2-approximation, that is, the computed value is $\geq D/2$, where D is the graph diameter.

Let G be the graph we are performing BFS on and V the set of its vertices.

For our proof, we make use of the following property of BFS.

Remark. *Let a be a vertex in G. Let T be the breadth-first tree rooted at a. For every vertex v in G, the length of the shortest path between a and v in G is the length of the path from a to v in T.*

Proof. We pick a vertex a to start BFS and construct the tree T.

Now we take the furthest vertex from a in T that is

$$v = \underset{x \in V}{\operatorname{argmax}}(d(a, x))$$

We know from triangle inequality that

$$\forall (u, w) \in V.\ d(u, w) \leq 2 \times d(a, v)$$

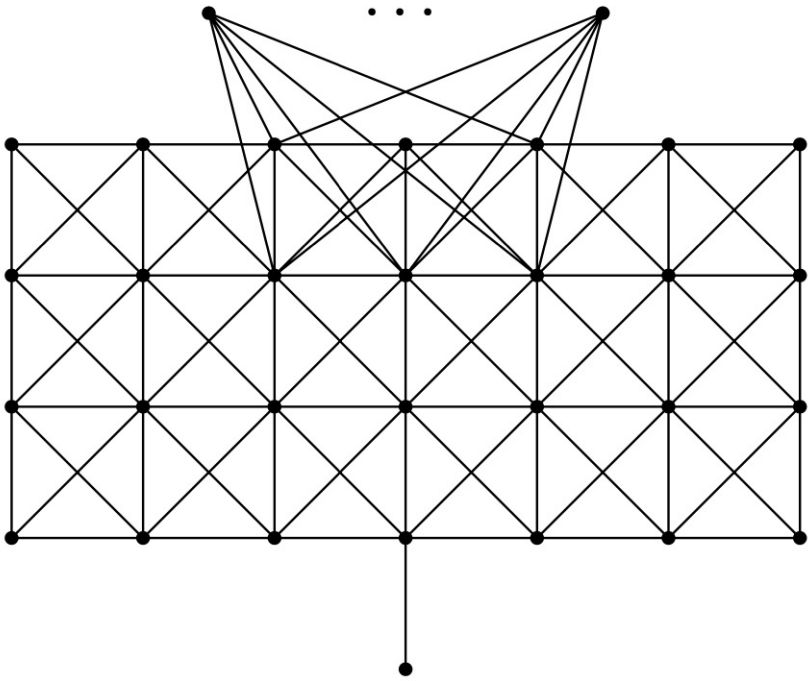
We proved the length of the longest shortest path is between $d(a, v)$ and $2 \times d(a, v)$ and this is the definition of diameter thus proving our thesis

$$d(a, v) \leq D \leq 2 \times d(a, v) \implies d(a, v) \geq D/2.$$

□

Exercise 3

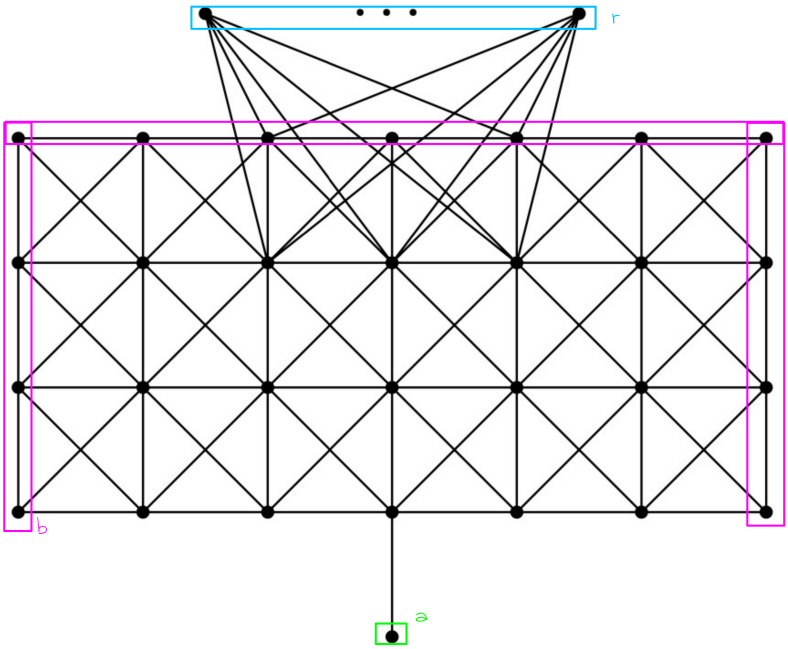
Consider the 2-sweep algorithm in an unweighted graph. Prove that this algorithm, which performs quite well in practice, gives an arbitrary lower value with respect to the diameter D of the following graph:



The algorithm is defined as:

```
def 2-Sweep(G):
        r ← random node in G;
        a ← farthest node from r;
        b ← farthest node from a;
        return d(a, b);
```

First, we consider the graph represented in the picture above. We can see the diameter is 6, but if we can force the algorithm to choose node r from the row above the grid, we get the one on the bottom as a, and the farthest nodes from this a are distant 4 which is not the diameter (refer to the picture in the following page).



Now that we have roughly explained our logic, we can generalize.

Theorem. *Let G be a graph made up of a grid of $n \geq 2$ rows, $2 \times n - 1$ columns as in the pictures above. If the starting random node r is chosen from those in the blue rectangle, the 2-sweep algorithm returns n.*

Proof. We can prove this by induction on k. □

The number of nodes in the blue rectangle can be chosen such that the probability of a random node to be chosen in the graph to be in the blue rectangle is arbitrarily high. Calling n the number of nodes in the blue rectangle, we can define the probability of choosing a random node r among these:

$$\begin{aligned}
&\Pr[r \text{ is in the blue rectangle}] \\
&= 1 - \Pr[r \text{ is in the grid}] \\
&= 1 - \frac{k \times (2k-1)}{k \times (2k-1) + n}
\end{aligned}$$

□