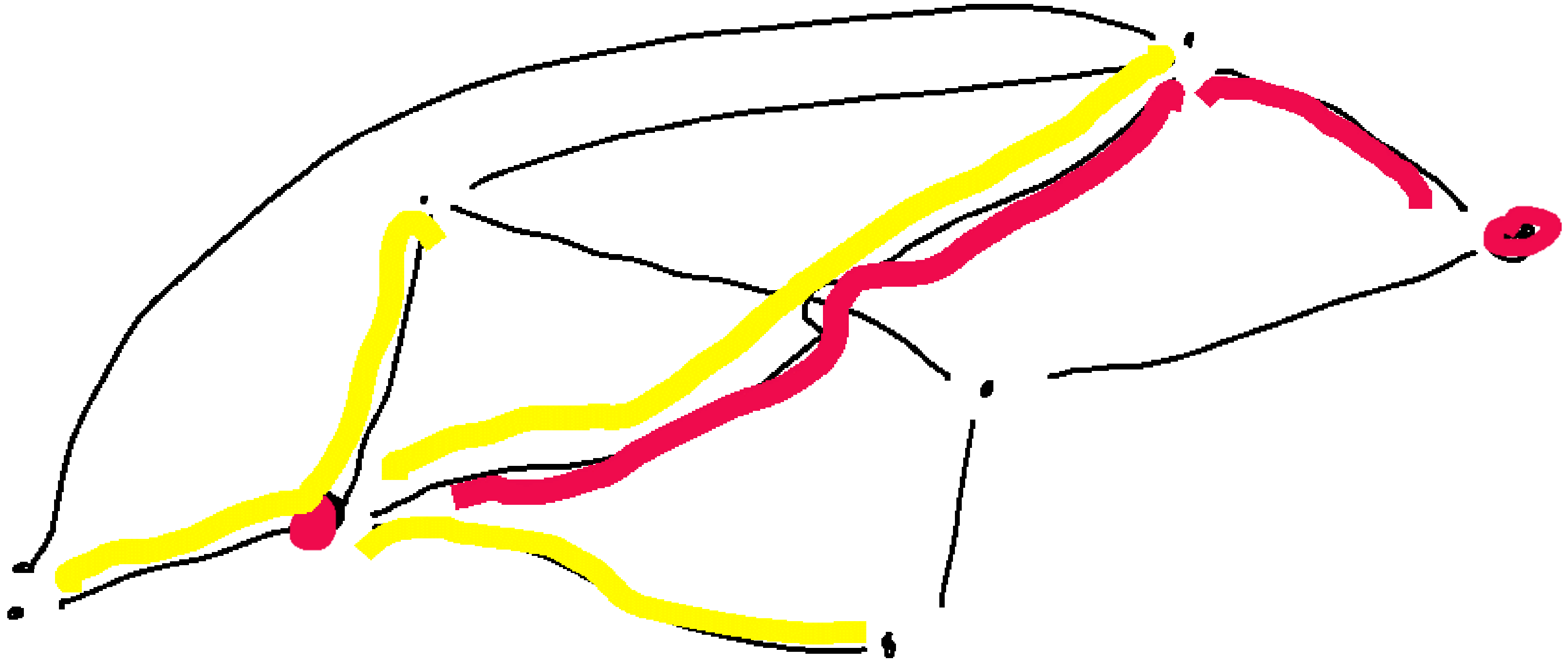
Communication complexity

Algo 21

All pairs shortest path problem (APSP)

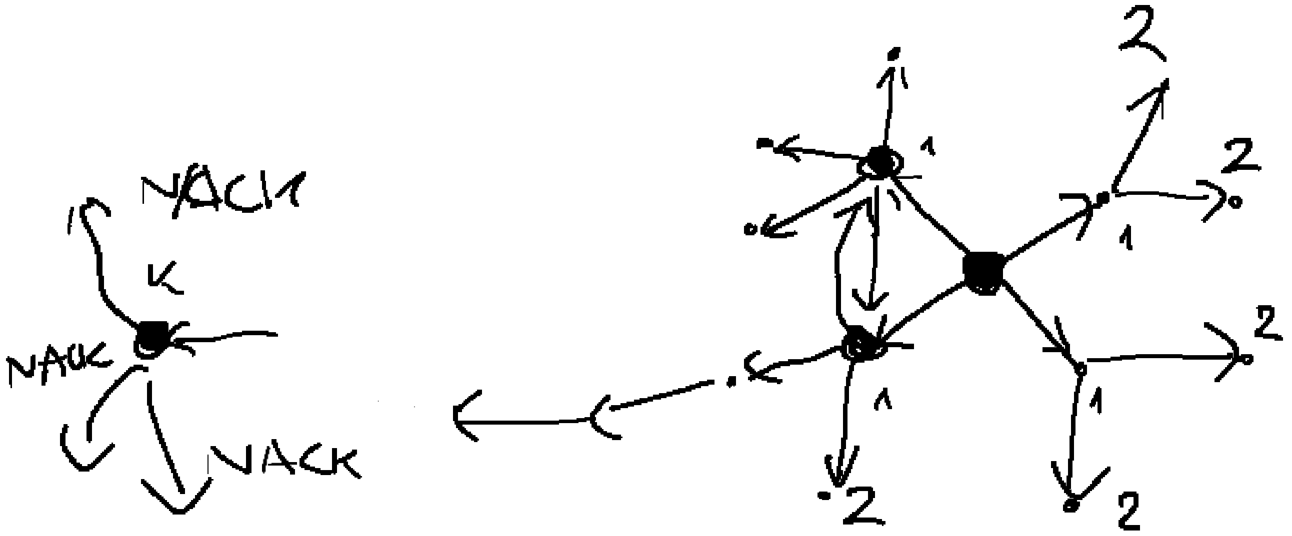
problem



Naïve solution

Algorithm 11.1 Naive Diameter Construction

all nodes compute their radius by synchronous flooding/echo
all nodes flood their radius on the constructed BFS tree
the maximum radius a node sees is the diameter

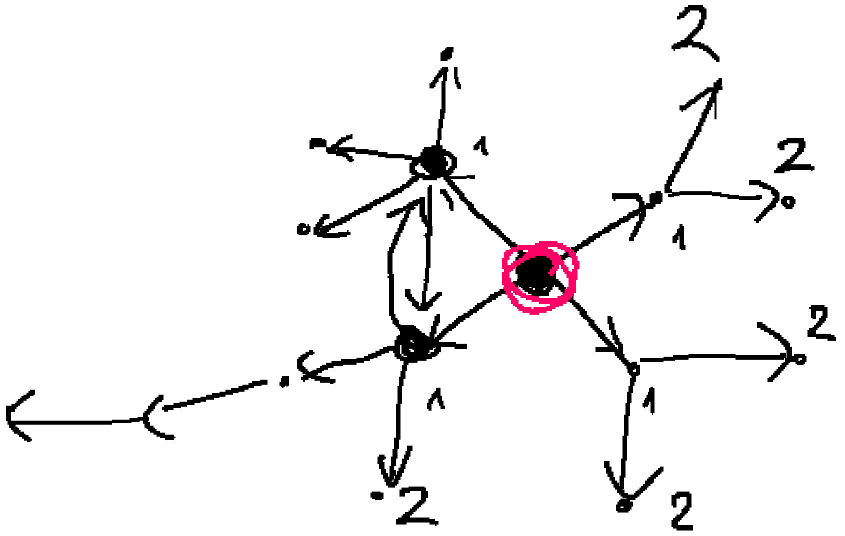


1st time: ACK
¬1st time: NACK

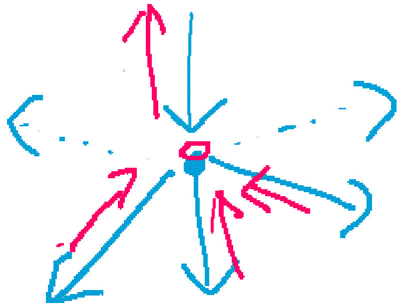
Naïve solution

Algorithm 11.1 Naive Diameter Construction

~~all nodes compute their radius by synchronous flooding/echo~~
all nodes flood their radius on the constructed BFS tree
the maximum radius a node sees is the diameter



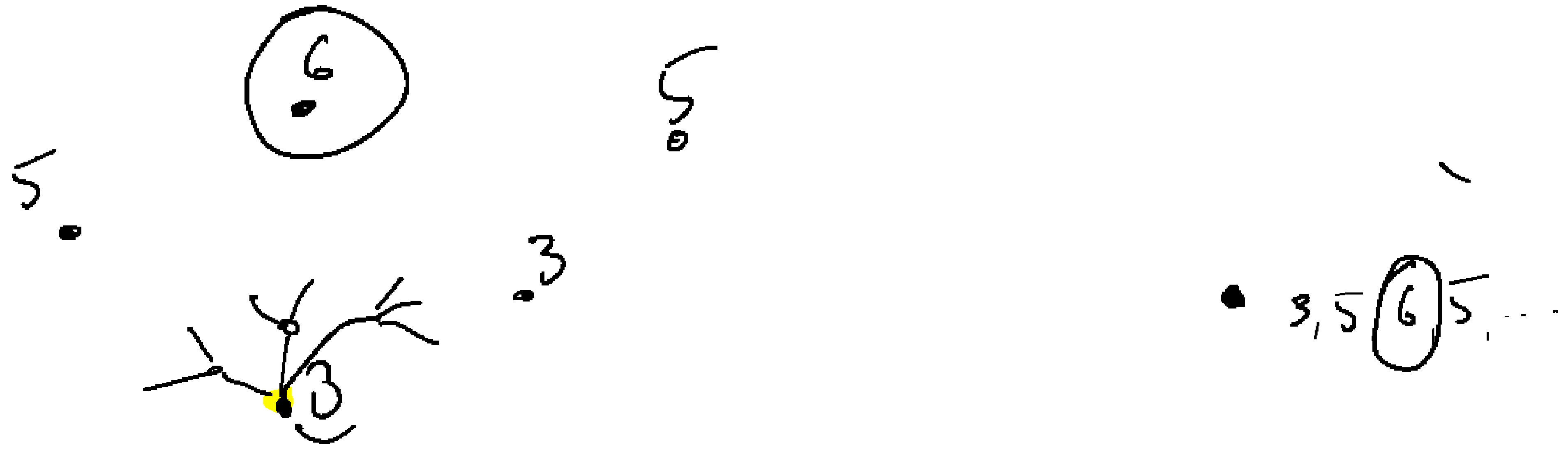
1st time: ACK
¬1st tre: NACK



Naïve solution

Algorithm 11.1 Naive Diameter Construction

all nodes compute their radius by synchronous flooding/echo
all nodes flood their radius on the constructed BFS tree
the maximum radius a node sees is the diameter

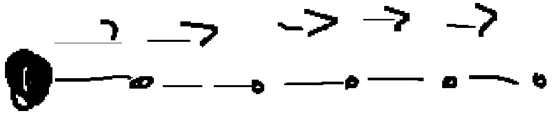


Naïve solution complexity

2 log n

time O(D) for diameter D → O(n · log n)

Congestion of messages – n algorithms executed in parallel



D

time:

1 flood + echo $O(D)$

2 BFS trees $O(D)$

Congestion

n messages at once

$\geq n \log n$

processing: n steps

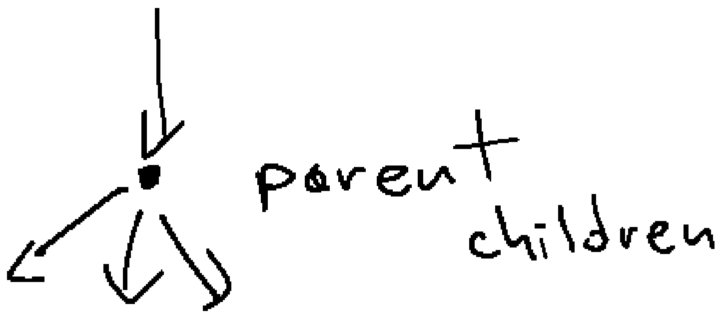
Reasonable size of messages

something like

ID for a node
is of length $\log(n)$

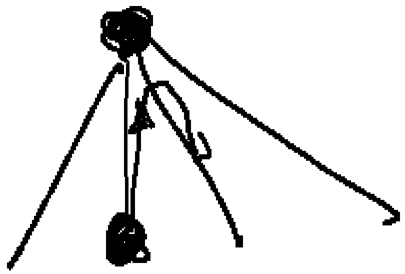
Building block -- BFS

Definition 11.2. *(BFS_v) Performing a breadth first search at node v produces spanning tree BFS_v (see Chapter 2). This takes time $\mathcal{O}(D)$ using small messages.*



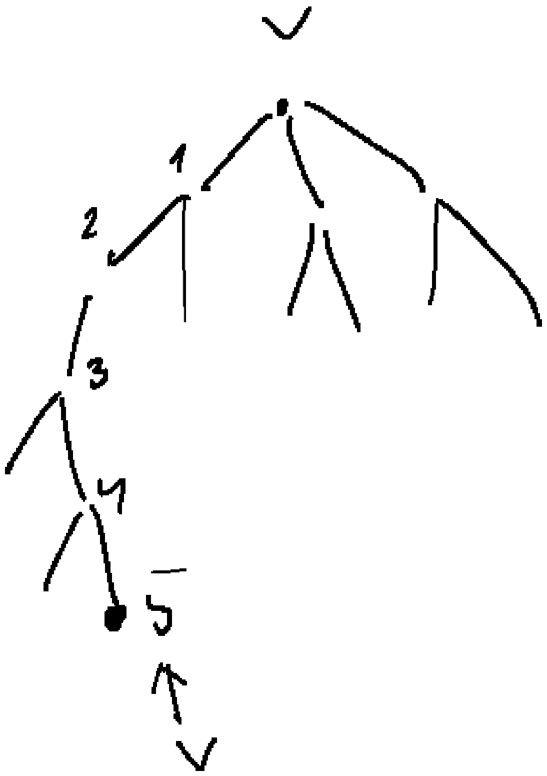
DFS based on BSF

Pebbles algorithm



Algorithm 11.3 Computes APSP on G.

```
Assume we have a leader node ~~l~~ (if not, compute one first)
~~compute BFS_l of leader l~~
~~send a pebble P to traverse BFS_l in a DFS way;~~
while P traverses BFS_l do
    if P visits a new node v then
        wait one time slot;          // avoid congestion
        start BFS_v from node v      // ~~compute all distances to v~~
        // the depth of node u in BFS_v is d(u, v)
    end if
end while
```



Algorithm 11.3 Computes APSP on G.

```
Assume we have a leader node l (if not, compute one first)
compute BFS_l of leader l
send a pebble P to traverse BFS_l in a DFS way;
while P traverses BFS_l do
    if P visits a new node v then
        wait one time slot;   // avoid congestion
        start BFS_v from node v;   // compute all distances to v
        // the depth of node u in BFS_v is d(u, v)
    end if
end while
```

Algorithm 11.3 Computes APSP on G.

```
Assume we have a leader node l (if not, compute one first)
compute BFS_l of leader l
send a pebble P to traverse BFS_l in a DFS way;
while P traverses BFS_l do
    if P visits a new node v then
        wait one time slot;   // avoid congestion
        start BFS_v from node v;   // compute all distances to v
        // the depth of node u in BFS_v is d(u, v)
    end if
end while
```

Avoiding congestions

Lemma 11.4. *In Algorithm 11.3, at no time a node v is ~~simultaneously active~~ for ~~both~~ $\cancel{BFS_u}$ ~~and~~ $\cancel{BFS_v}$.*

- Let: BFS started at u at time t(u), at v at time t(v) $\quad t(u) \neq t(v)$ because of P
- node v involved at time t(u)+d(u,v), so t(v)>= t(u)+d(u,v)+1
- $$\underbrace{t_v + d(v,w)}_{BFS_v \text{ on } w} \geq (t_u + \cancel{d(u,v)}) + 1) + \cancel{d(v,w)} \geq t_u + \cancel{d(u,w)} + 1 > t_u + d(u,w)$$

Time complexity

Theorem 11.5. *Algorithm 11.3 computes APSP (all pairs shortest path) in time* $O(n)$.

pebble time: $O(n)$

BFS_1 $O(D)$

~~last~~ BFS_n $O(D)$

Time: $O(D) \to O(n)$

$O(D \cdot n)$ → message $\log(n)$ and not $n \log n$

Lower bound

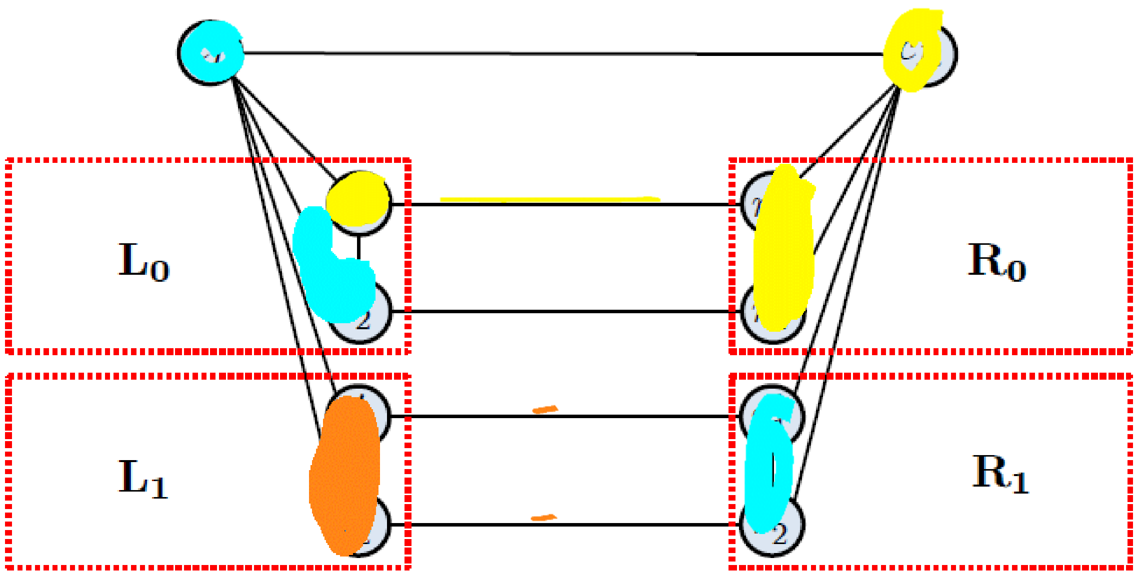
time $O(n/\log n)$

Graph used for showing n/log n lower bound

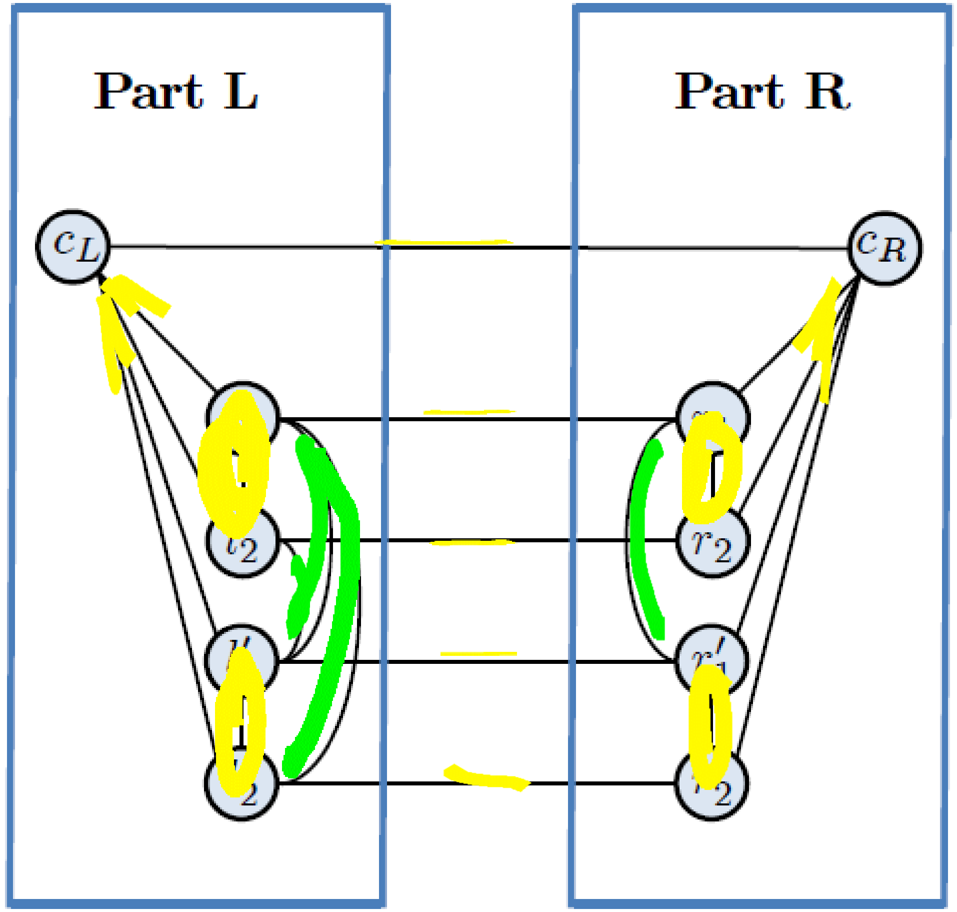
$$\begin{array}{llll} \mathbf{L_0} & := & \{l_i \mid i \in [q]\} & \text{// upper left in Figure 11.6} \\ \mathbf{L_1} & := & \{l'_i \mid i \in [q]\} & \text{// lower left} \\ \mathbf{R_0} & := & \{r_i \mid i \in [q]\} & \text{// upper right} \\ \mathbf{R_1} & := & \{r'_i \mid i \in [q]\} & \text{// lower right} \end{array}$$

$[q] = \{1, 2, \ldots, q\}$

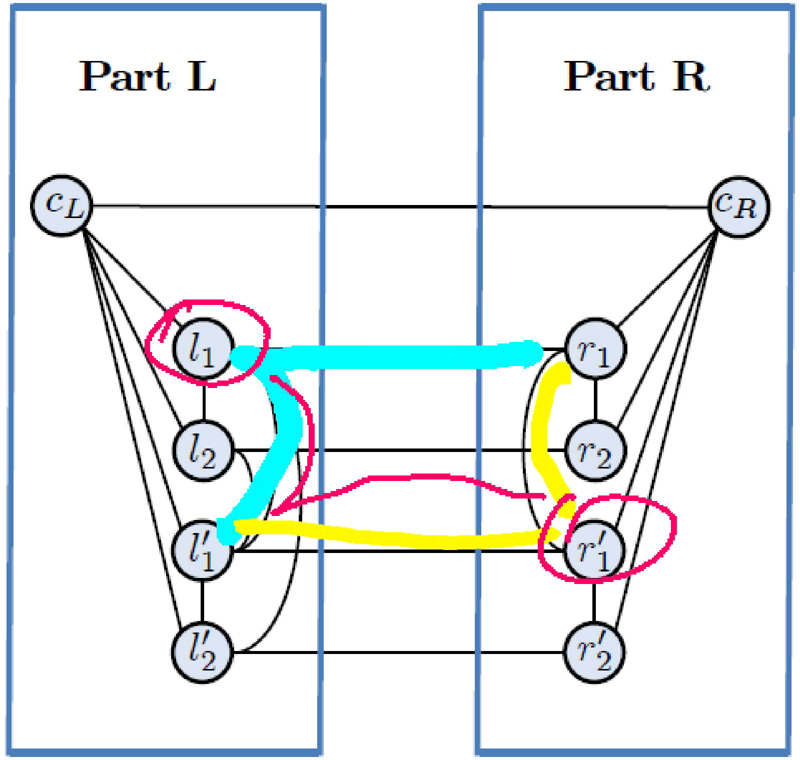
Clique of q nodes



Clique


Part L
Part R
c_L
c_R
l_2
r_2

Diameter 2 or 3



$\forall i, j$

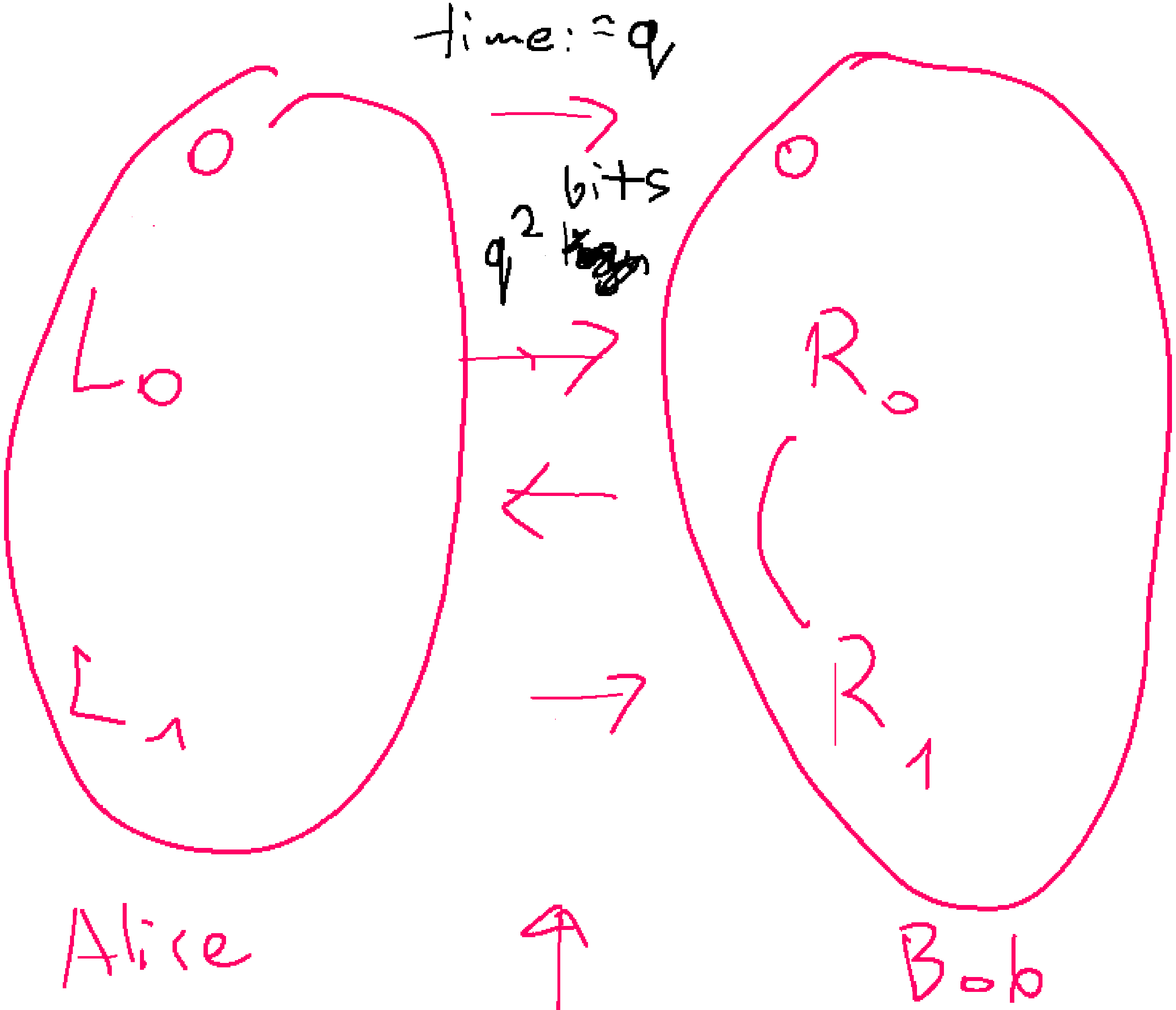
l_i connected to r_i'

or

l_i' connected to r_i

Cutsize

#nodes:
$4q+2$



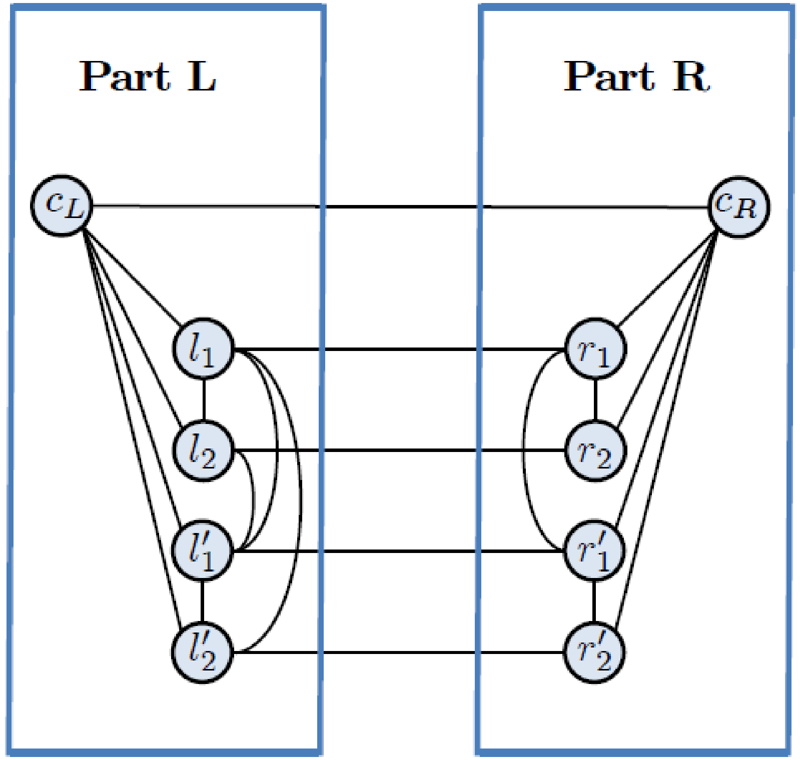
$i,j \Rightarrow$
$\exists$ edg

Add. edges: q^2 places

$2q+1$ edges

q^2 places

Cutsize



x, y

$x[i]=0$

$\Rightarrow y[i]=1$

$y[i]=0 \Rightarrow x[i]=1$

Informal argument

2-party communication model

Alice $\quad\quad\quad\quad\quad\quad\quad\quad$ Bob

x $\quad\quad\quad\quad\quad\quad\quad\quad$ y

$|x| = n$ $\quad\quad\quad\quad\quad\quad\quad\quad$ $|y| = n$

Boolean

to be learnt: $f(x, y)$

complexity = # bits exchanged

$\leq n + 1$

1) Send x to Bob

2) Bob transmits $f(x, y)$

Communication complexity

optimal algorithm

maximum #bits for the worst input

Equality, its complexity?

(Equality.) We define the equality function EQ *to be:*

$$\mathrm{EQ}(x,y) := \begin{cases} 1 & : x = y \\ 0 & : x \neq y \end{cases} .$$

average might be small

$x[1] \rightarrow$ $\qquad$ $y[1] \overset{?}{=} x[1]$

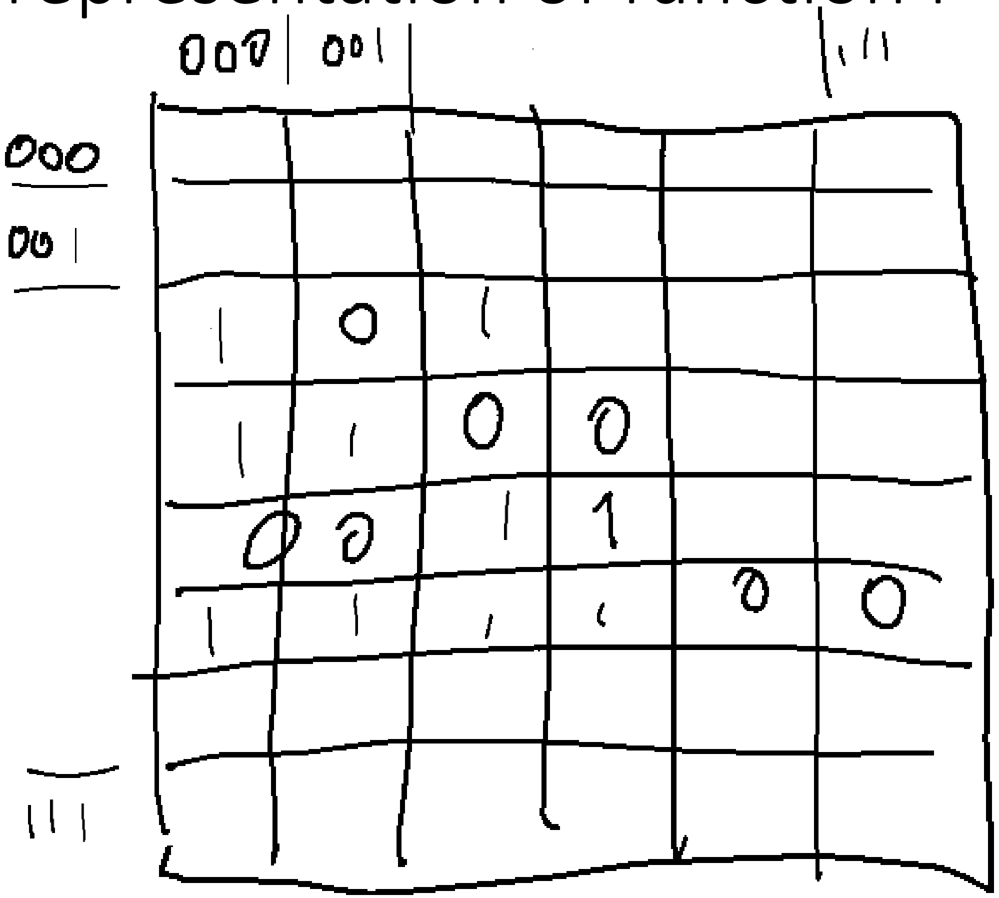
$x[2] \rightarrow$

$x[3] \rightarrow$

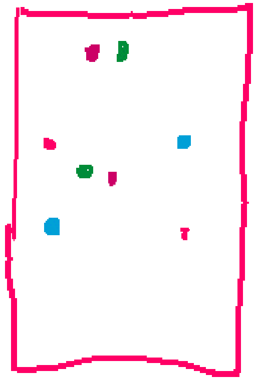
$\leftarrow$ NO

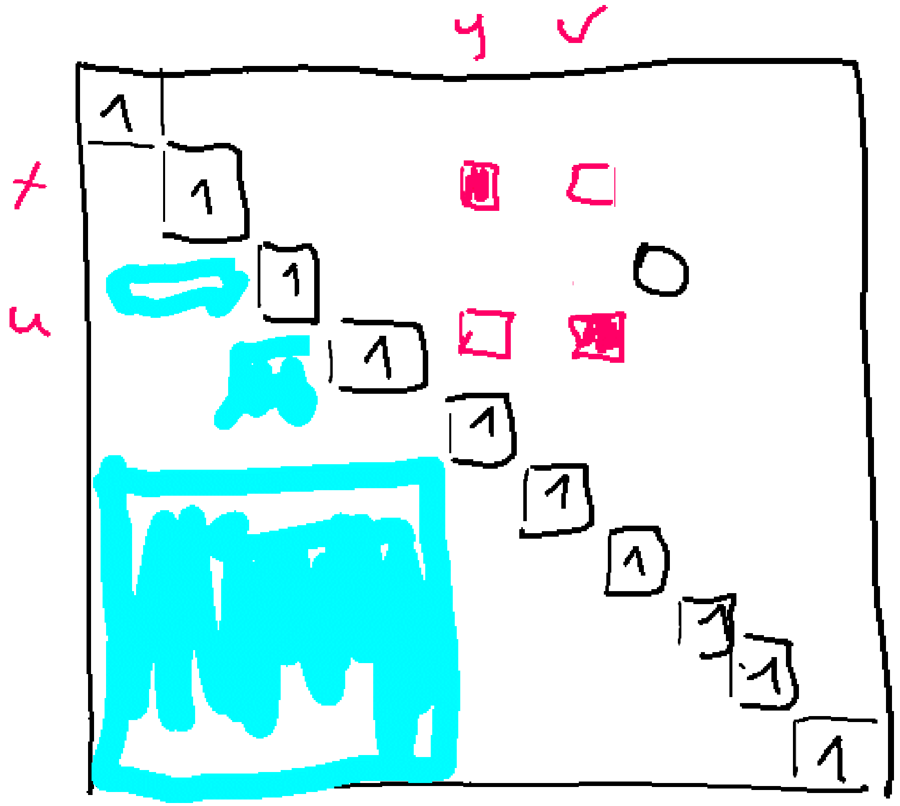
Formal definition of comm. complexity

Matrix representation of function f

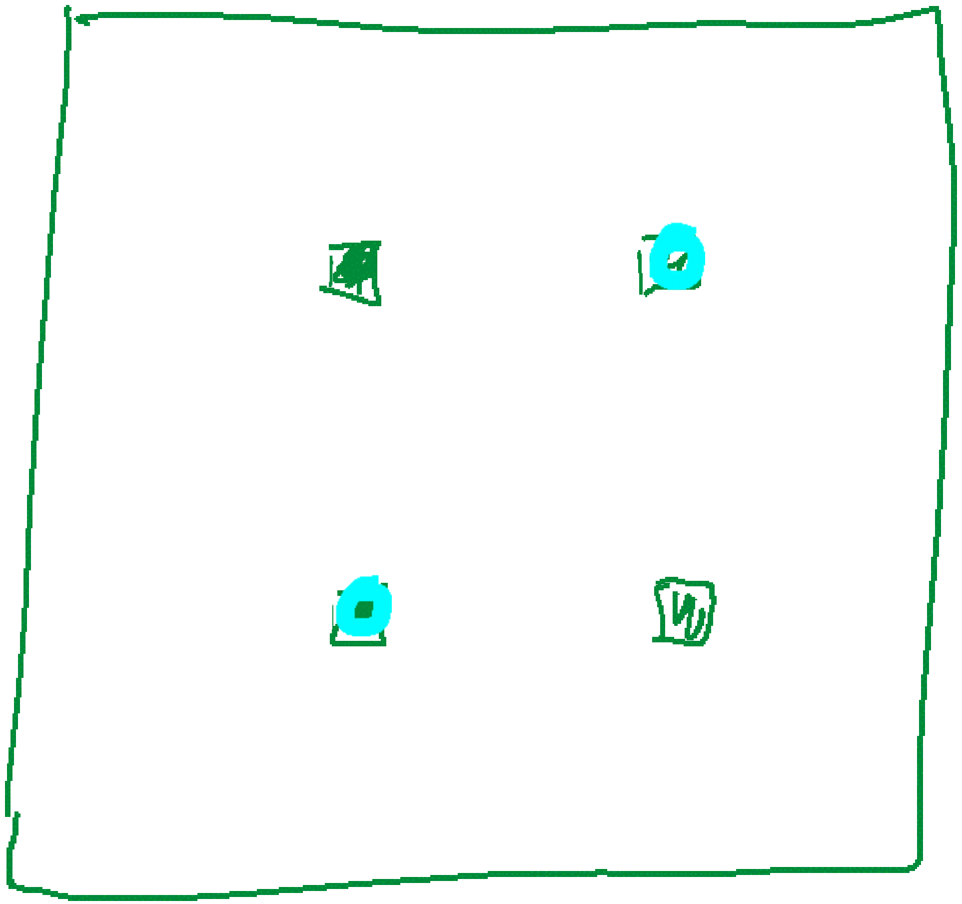


rectangles





$$(x,y) \in R$$
$$(u,v) \in R$$
$$\Downarrow$$
$$(x,v) \in R$$
$$(u,y) \in R$$



Monochromatic rectangles

EQ	000	001	010	011	100	101	110	111	$\leftarrow x$
000	1	0	0	0	0	0	0	0	
001	0	1	0	0	0	0	0	0	
010	0	0	1	0	0	0	0	0	
011	0	0	0	1	0	0	0	0	
100	0	0	0	0	1	0	0	0	
101	0	0	0	0	0	1	0	0	
110	0	0	0	0	0	0	1	0	
111	0	0	0	0	0	0	0	1	
$\uparrow y$									

Fooling set

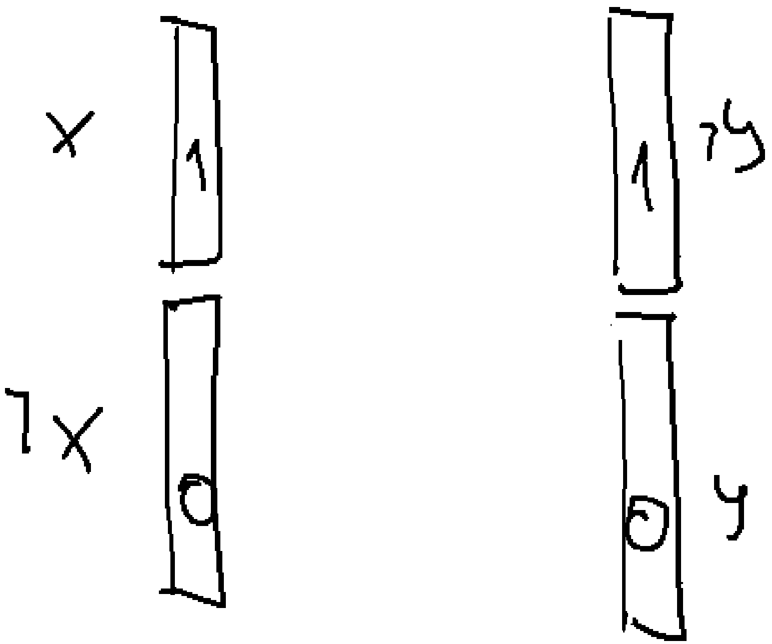
Fooling set lemma

If S is a fooling set for f, then $CC(f) = \Omega(\log |S|)$.

Equality

Auxiliary fact

Lemma 11.21. *Let x, y be k-bit strings. Then $x \neq y$ if and only if there is an index $i \in [2k]$ such that the i^{th} bit of $x \circ \overline{x}$ and the i^{th} bit of $\overline{y} \circ y$ are both 0.*



Mapping to graph

Definition 11.22. *Using the parameter q defined before, we define a bijective map between all pairs x, y of q^2-bit strings and the graphs in $\mathcal{G}$: each pair of strings x, y is mapped to graph $G_{x,y} \in \mathcal{G}$ that is derived from skeleton G' by adding*

- *edge (l_i, l'_j) to **Part L** if and only if the $(j + q \cdot (i-1))^{th}$ bit of x is 1.*
- *edge (r_i, r'_j) to **Part R** if and only if the $(j + q \cdot (i-1))^{th}$ bit of y is 1.*

x — length q^2

y —

q^2

Mapping to graph

x, y

Lemma 11.23. *Let x and y be $\frac{q^2}{2}$-bit strings given to Alice and Bob.*[1] *Then graph $G := G_{x \circ \bar{x}, \bar{y} \circ y} \in \mathcal{G}$ has diameter 2 if and only if $x = y$.*

$x \circ \bar{x}$

$\bar{y} \circ y$

diameter 2
iff no i
$x \circ \bar{x}$ and $\bar{y} \circ y$
has both ~~n~~ 0's

$x = y$

diameter 3
otherwise

$x \neq y$

Lower bound

in graph $\quad n$ ~~messages~~ bits exchanged

message = $\log n$ bits

$$\Rightarrow \#\text{messages} \geq \frac{n}{\log n}$$

Lower bound

Randomized complexity of equality

Algorithm 11.25 Randomized evaluation of EQ.

Alice and Bob use public randomness. That is they both have access to the same random bit string $z \in \{0,1\}^k$

Alice sends bit $a := \sum_{i\in[k]} x_i \cdot z_i \mod 2$ to Bob

Bob sends bit $b := \sum_{i\in[k]} y_i \cdot z_i \mod 2$ to Alice

if $a \neq b$ **then**

  we know $x \neq y$

end if

$$a := \sum x_i \cdot z_i \mod 2$$

$$b := \sum y_i \cdot z_i \mod 2$$