Mobile Agent Rendezvous

Mobile Agent

Robots

Drones

Vehicles

Outline

- MA Model

- Tokens
- RV Algorithms
  - Two MAs
  - Time/Memory Tradeoffs
  - Rendezvous with Ddetection

MA Model

Rendezvous

Leader Election

- Two Mobile Agents (MAs) initially located at arbitrary nodes of a graph. MAs can move along edges. same speed.

- **RV** (Rendezvous) means the MAs meet. It is a form of gathering and a very basic form of information exchange.

- **RVP** (Rendezvous) Problem.
  - Give an algorithm that enables the MAs to rendezvous regardless of their starting positions. may never terminate
  - The requirement is to cause rendezvous, if rendezvous is not possible rendezvous will never happen, and the algorithm will be running for ever!

- **RVD**: Rendezvous Problem with Detection
  - A rendezvous algorithm is required to detect whether or not rendezvous is possible and if it is to cause rendezvous.

Graph Model

2D Geometric Model

a b

3D Geometric Model

Gathering

- For more than two agents in a graph the problem is also known (referred to) as Gathering.

- $Gather(k)$ means that k MAs meet.
- $Gather(2)$ is the same as **RV**.

MAs vs Message Passing

- So far, in distributed computations, we had messages that were being sent along vertices of a network.
  - The nodes were the processors that enabled the distributed computation.

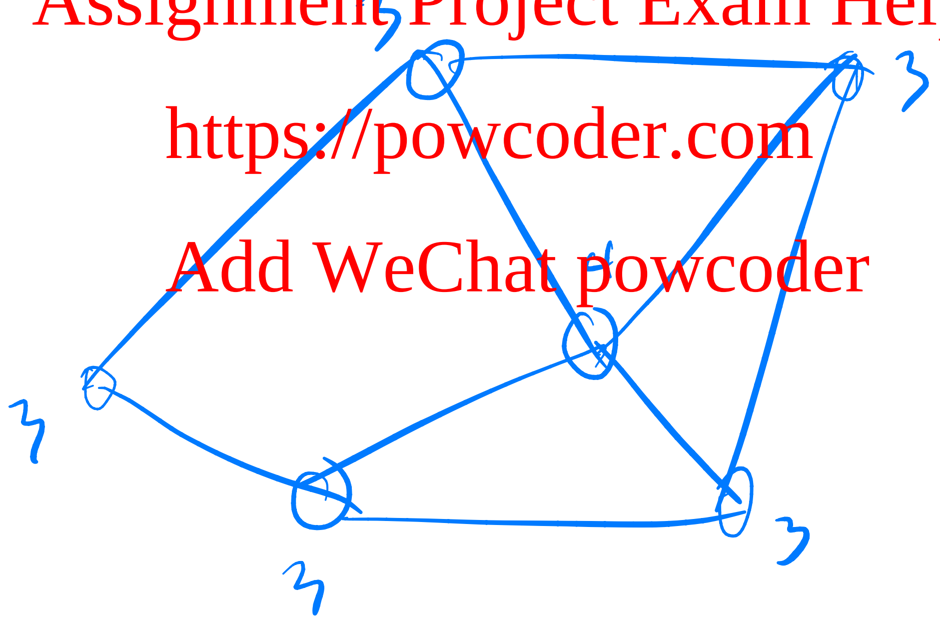
- This *distributed* setting remains, but now also
  - we have autonomous agents (or mobile code) moving on vertices of a distributed network
  - the agents are mobile; not only they can compute but also they can participate in distributed computations.

- The rendezvous problem can be stated for any topology.

Where/How can you rendezvous?

- Exploit network assymetries

- Exploit topological (geometric) assymmetries
- In a general graph the problem is quite complex!

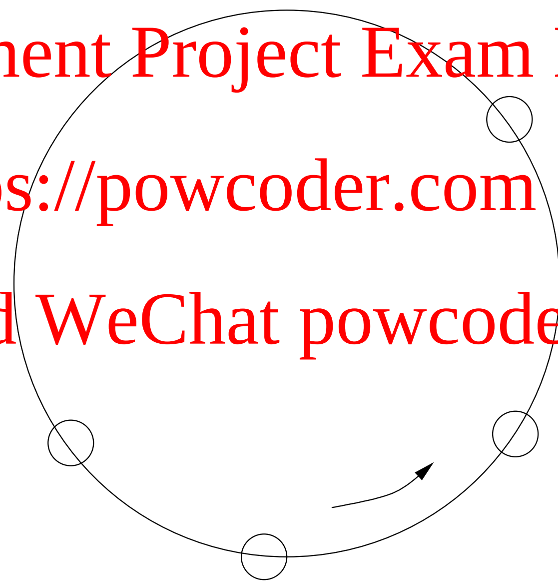


Special Topology: Ring

- We look only at ring topologies.

- Even in a ring the problem is quite complex!
- k MAs are initially located on k nodes of an n node ring.

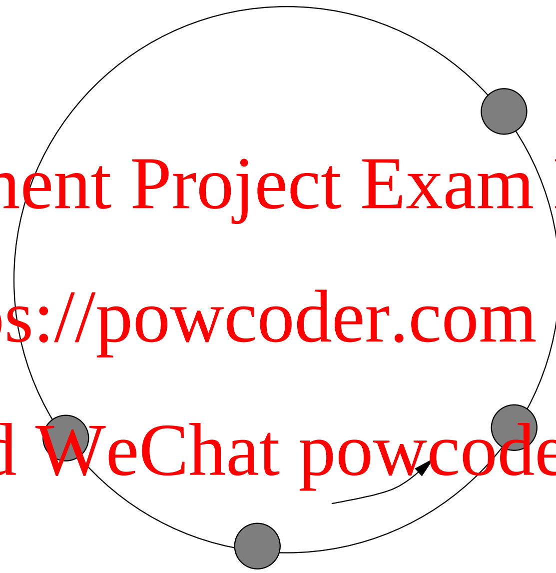
$k=2$

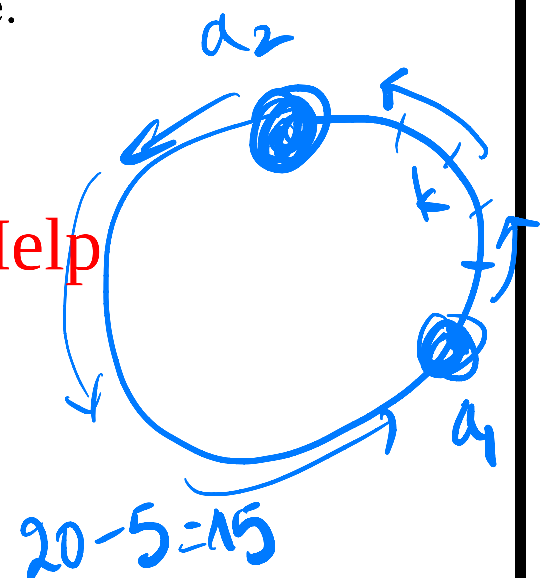


- Assume a synchronous system with common sense of direction.
- Will be restricted to the case of $k = 2$ MAs.

Ring, Mobile Agents, and Tokens

- Each of the MAs has a token which it chooses to leave (or not to leave) at its starting position or elsewhere.





Every agent owns its own token and can release it at any node; the tokens are indistinguishable

Approaches to Rendezvous (1/2)

- In general, to solve the rendezvous problem, one must try to break the inherent symmetry of the problem.

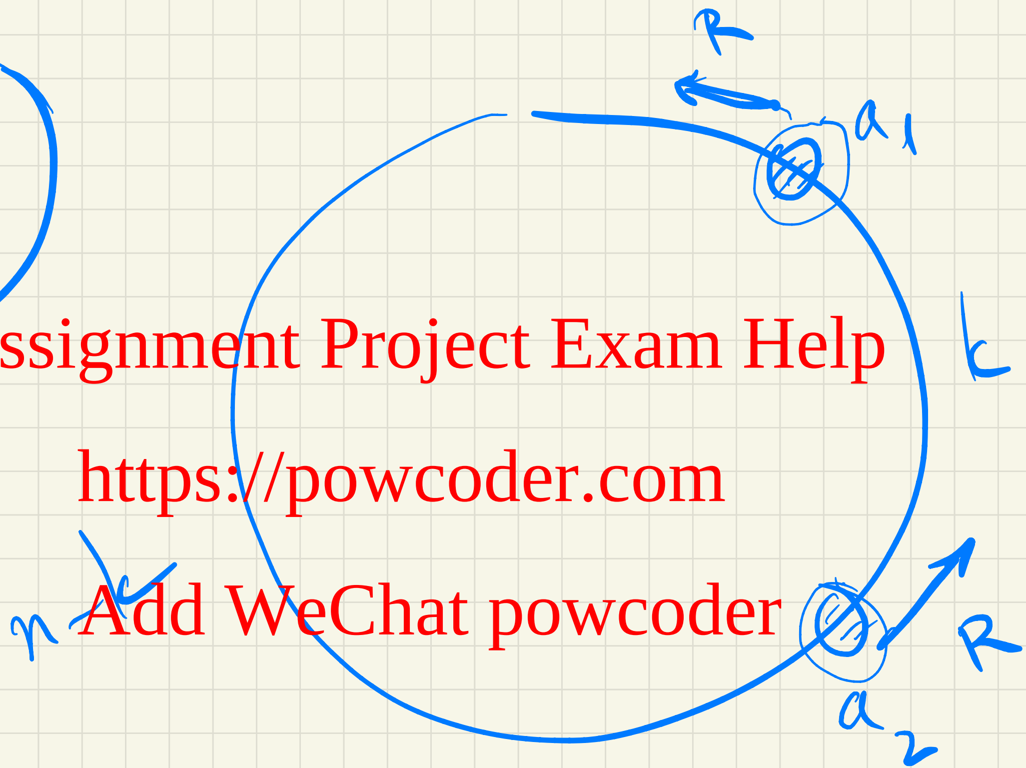
1. If the MAs have unique IDs you can break symmetry and the problem becomes, sometimes, easier to solve!
2. If the MAs do not have unique IDs and the network nodes are anonymous the resulting symmetry makes the problem difficult and sometimes even impossible to solve!

- In principle, we have to break symmetry:

  - either deterministically, or
  - using randomization.

- Sometimes we can use the underlying topology to break symmetry.

$k < \frac{n}{2}$



Approaches to Rendezvous (2/2)

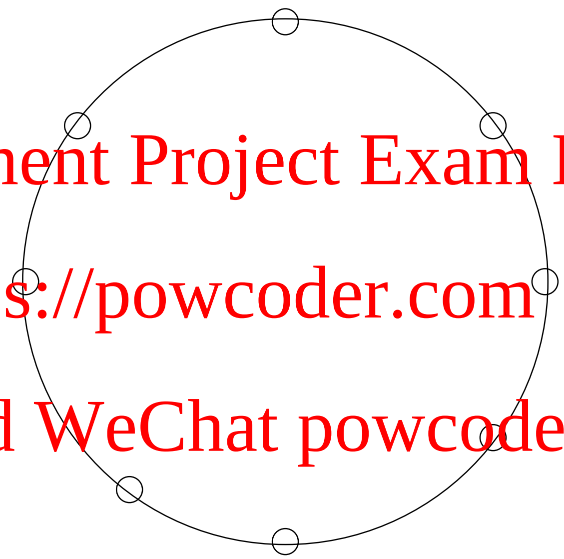
- Mobile agents have **identical** (i.e., indistinguishable) tokens, one token per MA, that they can release at their starting position.

- We are interested in tradeoffs among
  1. number of tokens
  2. knowledge
  3. time
  4. memory.

The Model: Topology and the Nodes

- There are n identical nodes on which MAs may reside.

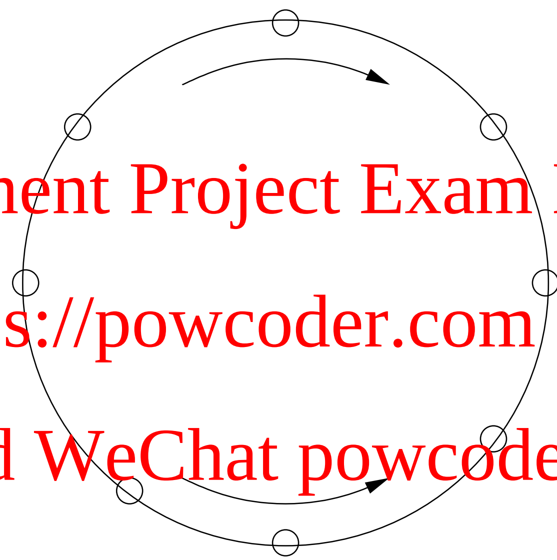
- The topology is a ring.



- For example, in the ring above there are $n = 8$ nodes and the MAs can traverse its vertices and edges.

The Model: Synchronicity/Orientation

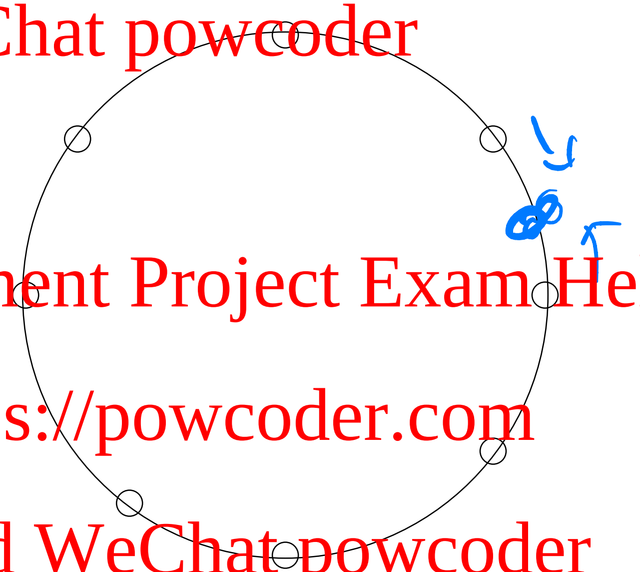
- The characteristics of the topology will be used in the algorithms.



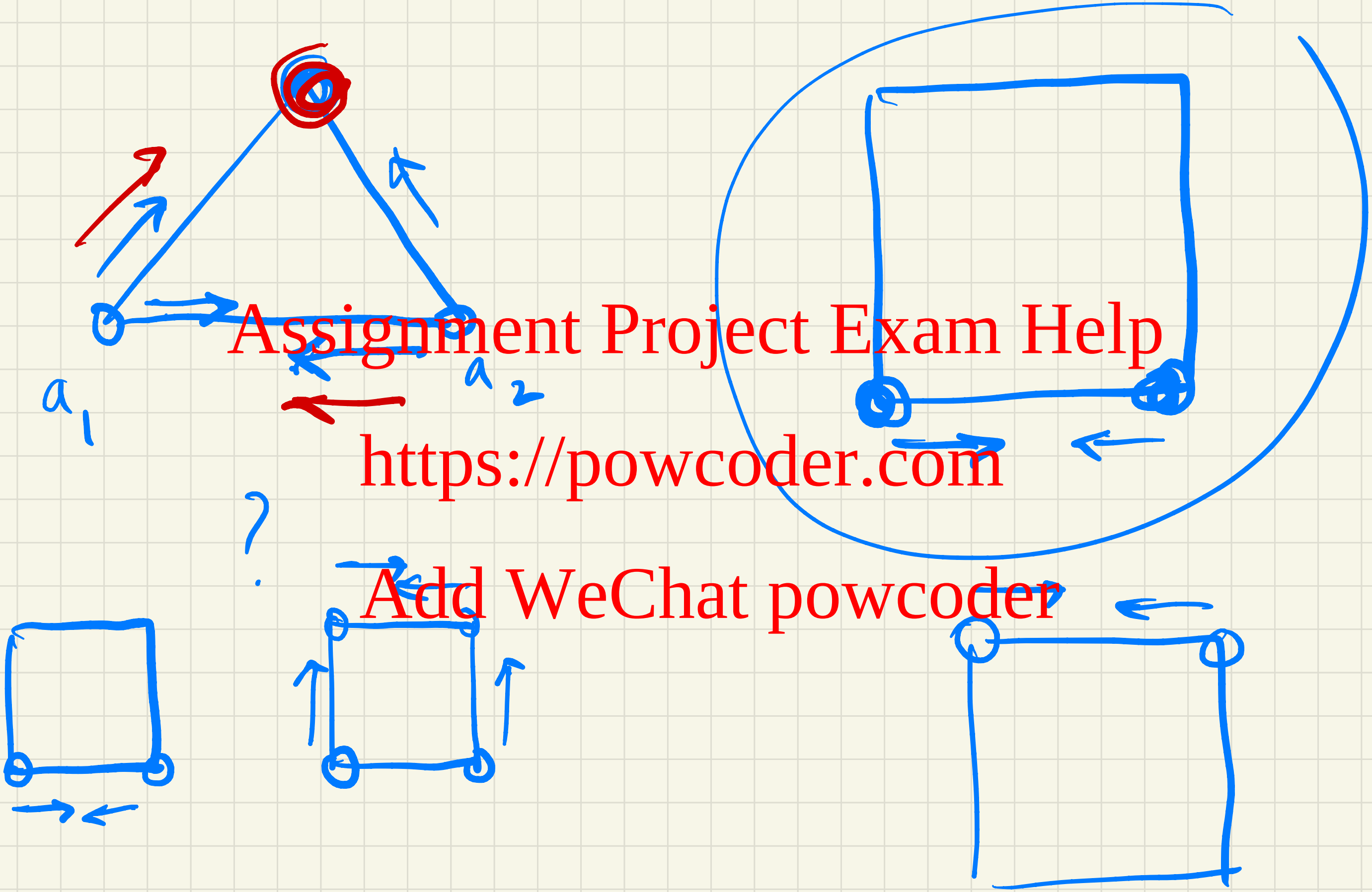
- Oriented means that there is a consistent sense of direction: either CW or CCW.
  - As usual this is specified with ports.

Where Can Rendezvous Occur?

- Can rendezvous for two agents occur at a node of the ring?



- Can rendezvous for two agents occur at an edge of the ring?
- Yes, to both questions!


a1
a2
?

Model

- Anonymous, synchronous, and possibly oriented n node ring.

- A given node requires only enough memory to host a token.
- Each MA, owns a single identical stationary token,
  - the tokens are indistinguishable and once they are positioned in the ring, they cannot be moved.

- A token or MA at a given node is visible by all MAs on the same node

  - The MAs follow the **same** deterministic algorithm and begin execution at the same time.
- Memory permitting, a MA can count the number of nodes visited, the number of nodes between tokens, or the total number of nodes in the network.

Impossibility of RV

Impossibility of RVD

- Consider two identical, anonymous Mobile Agents in a ring of size n.

- The agents have constant memory and can leave a token at their starting position.

- **Theorem 1** *No deterministic algorithm exists such that the MAs always correctly detect if* $d = \frac{n}{2}$ *and act appropriately, i.e., stop if* $d = \frac{n}{2}$ *and rendezvous otherwise.*

If one robot can move with speed 1 and the other with speed 2!

$d = \frac{n}{2}$

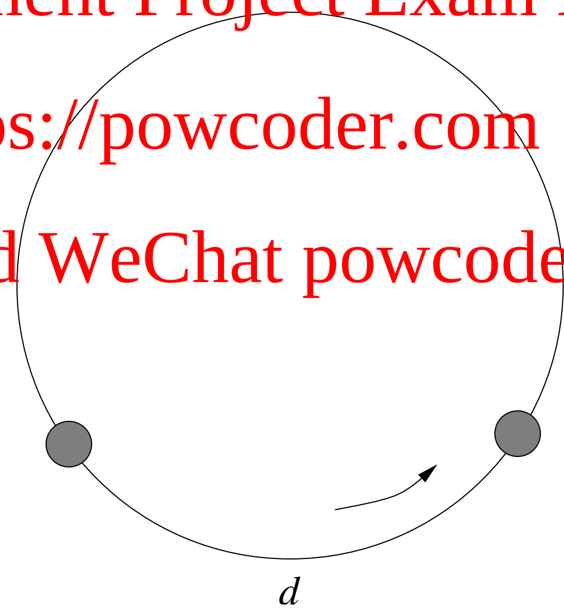
RV Algorithms

(Two Agents)

Rendezvous for two MAs

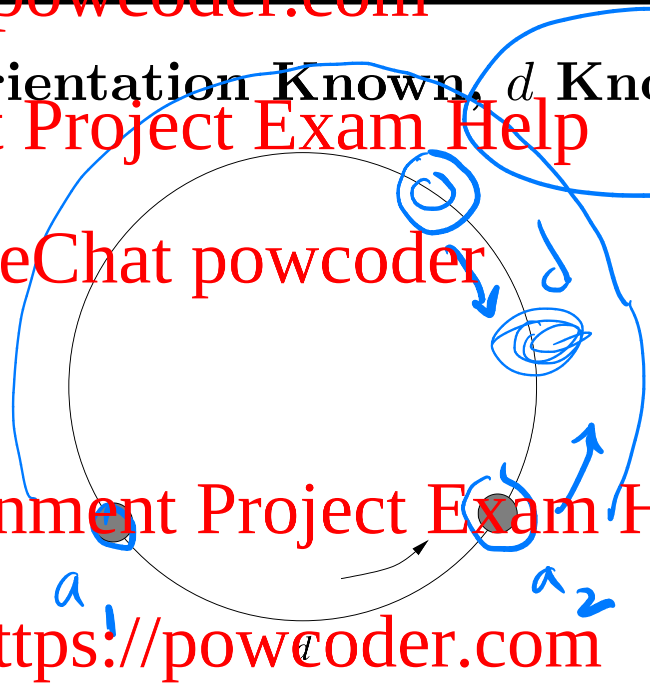
- RV in a ring

1. There are two **identical** MAs at distance d from each other.
2. The ring may or may not be oriented.
3. The ring has size n.



- There are a lot of (simple) subtleties!

1. Orientation Known, d Known



$d + \frac{d}{2}$

$n - d > d$

1. Release the token at starting position.
2. Walk around ring in a counterclockwise direction.
3. If a token found within d steps, continue in same direction.
4. If no token found by d steps, reverse direction and continue.

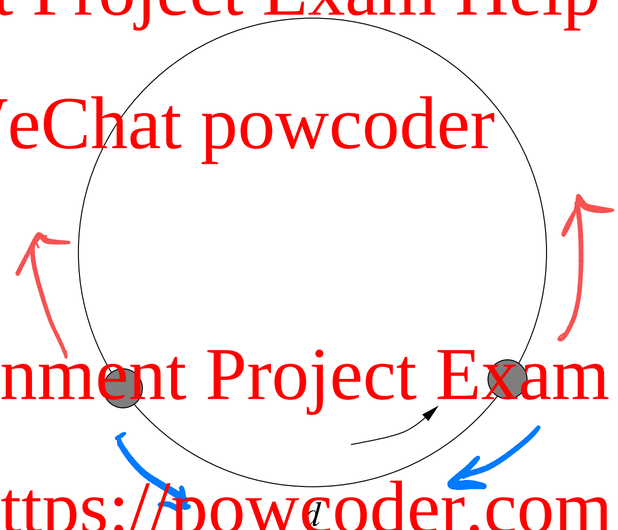
Correctness

- There are two "competing distances" measured by the MAs:
  - d and $n - d$.

- Rendezvous is achieved if $d < n/2$.
  - What if $d = n/2$?

- How many steps does it take to rendezvous?

2. Orientation not Known, d Known



$\frac{d}{2}, \frac{3d}{2}$

$\max\left\{\frac{d}{2}, \frac{3d}{2}\right\}$

1. Release the token.
2. Choose a direction and begin walking around the ring.
3. If a token is found within d steps, walk in the same direction.
4. If no token is found by d steps, reverse direction and continue.

$$\max\left\{\frac{n-d}{2}, \frac{3(n-d)}{2}, \frac{d}{2}, \frac{3d}{2}\right\}$$

Correctness

- Similar to previous algorithm except MAs may select either direction.

- There are two "competing distances" measured by the MAs:
  – d and $n - d$.

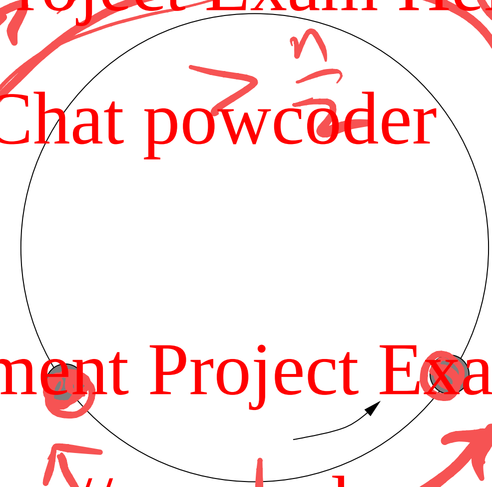
- Rendezvous is achieved if $d < n/2$.
  – What if $d = n/2$?

- How many steps does it take to rendezvous?

3. Orientation not Known, n Known

(initial distance is $< \frac{n}{2}$)

$> \frac{n}{2}$



$< \frac{n}{2}$

$O(\log n)$ memory

1. Release the token.
2. Choose a direction and begin walking around the ring.
3. If a token is found within $\frac{n}{2}$ steps, continue in same direction.
4. Otherwise, reverse direction at $\frac{n}{2}$ steps and continue.

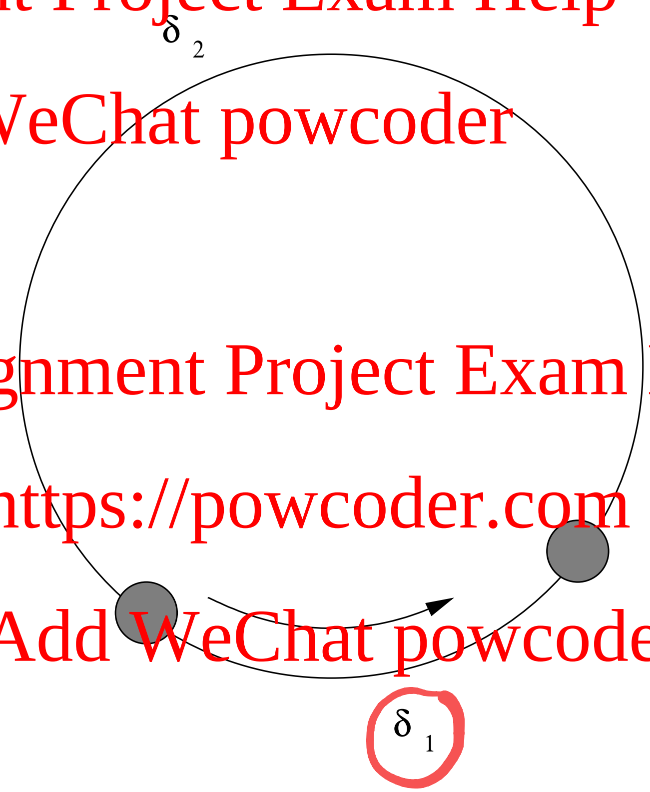
$d +$ (reverse trip)

Correctness

- Here algorithm does not need to know the distance.
  - $n/2$ is being used as a threshold for rendezvous.

- How many steps does it take to rendezvous?

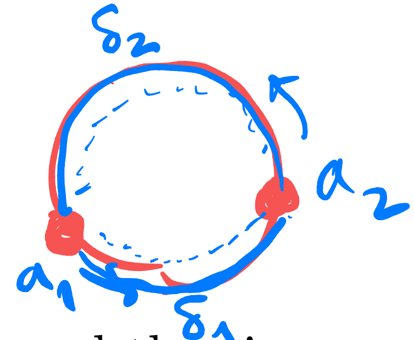
4. Orientation, d and n Are not Known.



Here the MAs must discover what is their distance!

Time/Memory Tradeoffs

Assume $O(\log n)$ memory



- Algprithm:

1. Release the token.
2. Choose a direction and begin walking around the ring.
3. Count # of steps to 1st token, δ_1, and continue.

4. Count # of steps to 2nd token, δ_2.

   /* The MA is back at its starting node.*/

5. If $\delta_1 < \delta_2$, continue walking in the same direction.

6. Otherwise, reverse direction and continue walking.

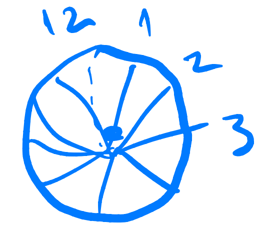
$\delta_1 \neq \delta_2$

- MAs need memory $O(\log n)$.
- Under what conditions on δ_1, δ_2 does the algorithm work?

a_1 sees (δ_1, δ_2) $\delta_1 < \delta_2$ stop

a_2 " (δ_1', δ_2') $\delta_2 = \delta_1' > \delta_2' = \delta_1$

Improving on the Memory



- A question remains:
  - can we improve on the memory $O(\log n)$?

- Knowledge of n requires $\log n$ bits.
  - How can we test using less than $\log n$ bits.
  - Answer: Use modular arithmetic!

modular arithmetic

- Previous algorithms were breaking symmetry by testing whether or not $d < n - d$.
  - We will find a way to carry out the same test, but we will generate prime numbers to do so!
  - In fact, it will be a sequence of tests: in the end it will be confirmed which of $d < n - d$ or $n - d < d$ is valid

11,054 : 45

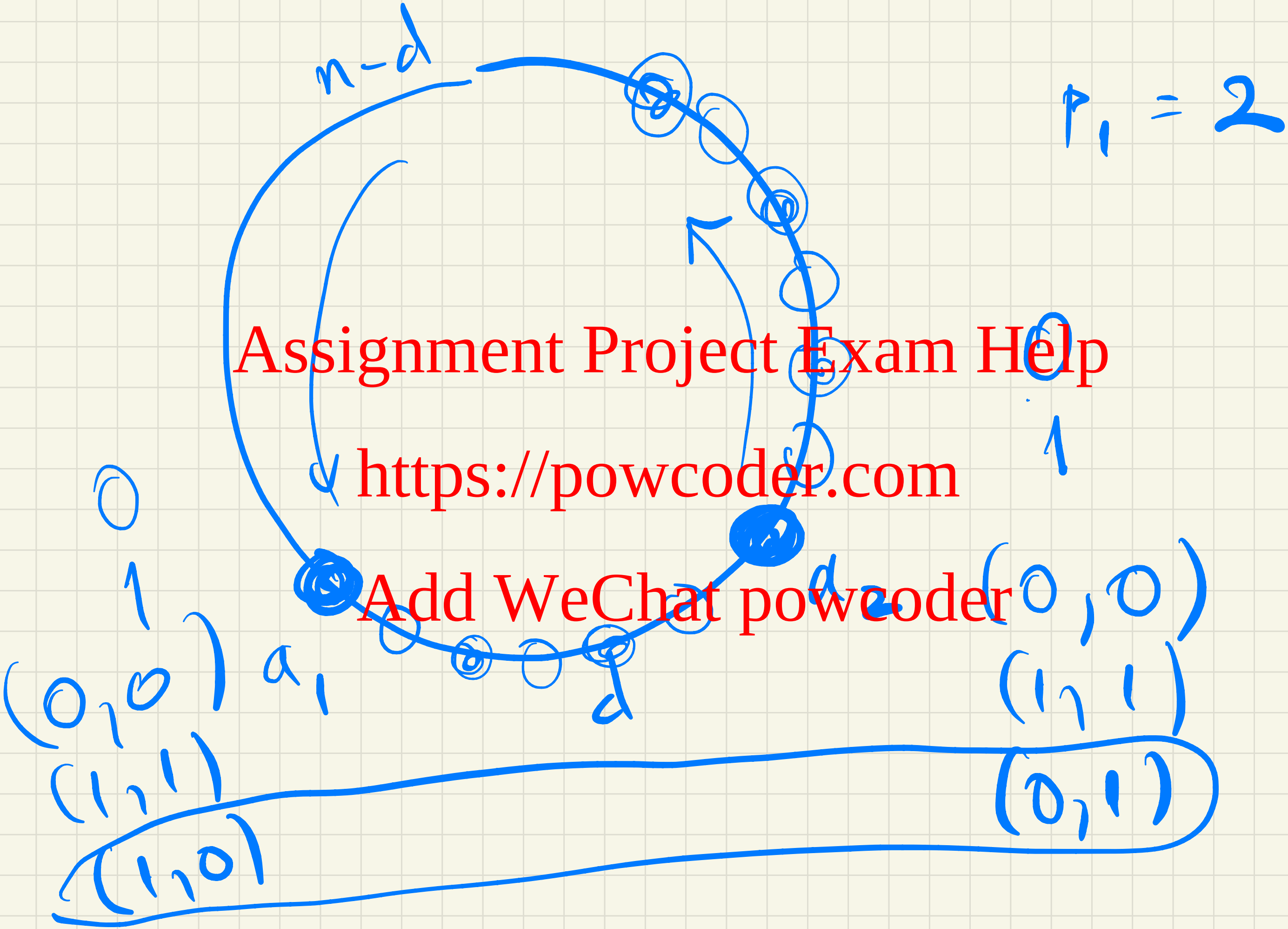
11,055 : 45

Algorithm

if $d < n-d$

1. Release the token.
2. Set $m = p_1$.
3. Choose a direction and begin travelling around the ring.
4. Count # of steps, mod$m$, to the first token, δ_1, and continue walking.
5. Count the number of steps, mod$m$, to the second token, δ_2. /* The MA is back at its starting node. */
6. If $\delta_1 \bmod m = \delta_2 \bmod m$, set $m = p_{i+1}$ and repeat from step 4.
7. If $\delta_1 \bmod m < \delta_2 \bmod m$, continue travelling in same direction.
8. Else, reverse direction and continue travelling.

- All we are doing is trying to test whether or not $d < n - d$, but modulo consecutive primes!
- MAs need to generate: $p_1, \ldots, p_k$ first k primes s.t. $\prod_{i=1}^{k} p_i > n$

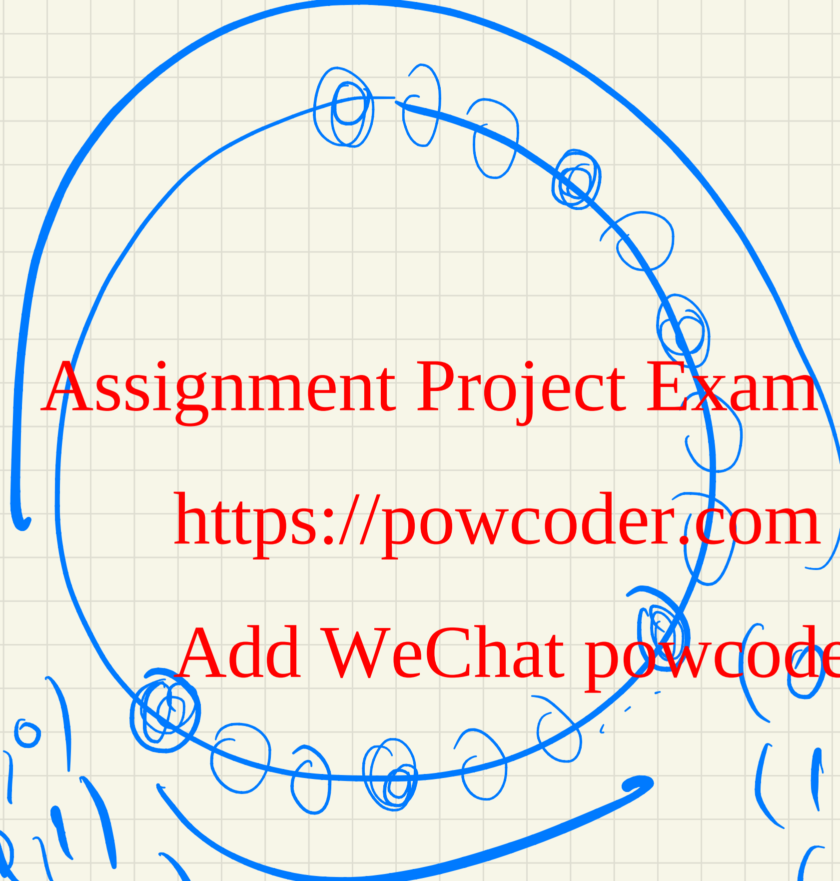

n-d
p1 = 2
0
1
d
a1
(0,0)
(1,1)
(1,0)
(0,1)

$p_2 = 3$

0

1

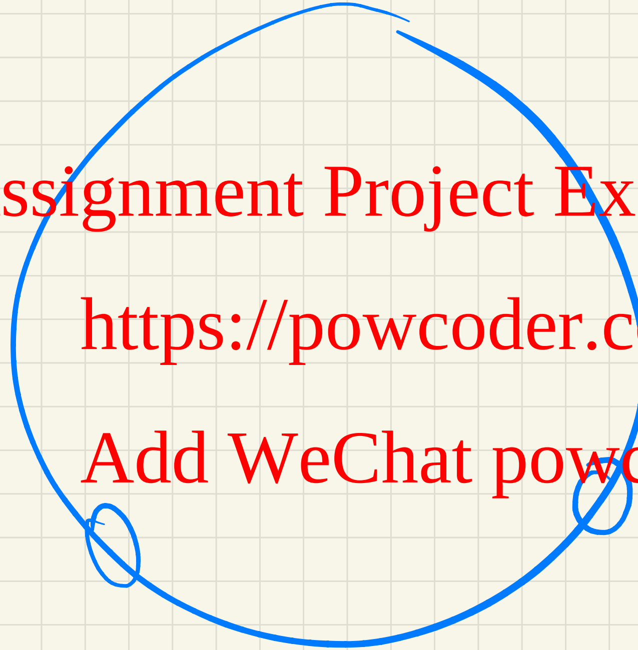
2



(0,0)
(1,1)
(2,2)
(1,2)

(0,0)
(1,1)
(2,2)
(2,1)

$P_3 = 5$



Prime numbers are the "building atoms" of numbers

~~1~~, 2, 3, 5, 7, 11, 13, 17, 19

Euclid (2,000 yrs ago) Alexandria

There are infinitely many primes!

Memory $O(\log \log n)$:

- **Theorem 2** *The previous algorithm requires Memory*

$$O(\log \log n)$$

$O(\log n)$

and accomplishes rendezvous in time

$$O\left(\frac{n \log n}{\log \log n}\right)$$

$O(n)$

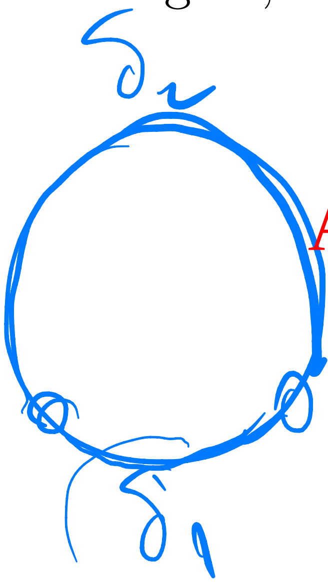
- Algorithm terminates after first k primes $p_1, \ldots, p_k$ s.t.

$$\prod_{i=1}^{k} p_i > n$$

- To prove termination use the Chinese Remainder Theorem.
- To prove time complexity use the Prime Number Theorem.

Using the Chinese Remainder Theorem[a]

- If the n_i are pairwise co-prime, and if $a_1, \ldots, a_k$ are any integers, then there exists an integer x such that



$$
\begin{aligned}
x &\equiv a_1 \bmod n_1 \\
x &\equiv a_2 \bmod n_2 \\
&\vdots \quad\quad \vdots \\
x &\equiv a_k \bmod n_k
\end{aligned}
$$

p_1 p_2 p_k

$p_1 p_2 \cdots p_k > n$

and x is unique $\bmod (n_1 n_2 \cdots n_k)$.

- In our case $n_i := p_i$, for $i = 1, 2, \ldots, k$, and there are the two solutions δ_1, δ_2.

I cannot do the details of this in this class!

[a] Problem with specific numbers, appears in the 3rd-century book Sunzi Suanjing by the Chinese mathematician Sunzi

Using the Prime Number Theorem

- The worst case occurs when $d = n/2$ since the max number k of prime numbers have to be checked.

- Resulting running time is $O(kn)$, but we need to determine the value of k. Consider smallest k such that $\prod_{i=1}^{k} p_i > n$.

- This implies that $\prod_{i=1}^{k-1} p_i \le n$. Hence $2^{k(k-1)/2} \le n$. Thus $p_k \approx k \ln k \le k^2 \le n$.[a]

$$\left|\{\underset{\text{prime}}{p} : p \le n\}\right| \approx \frac{n}{\ln n}$$

- Therefore $\prod_{i=1}^{k} p_i \le n p_k \le n^2$.

- By the prime number theorem we have that

$$n^2 \ge \prod_{i=1}^{k} p_i \ge \prod_{i=1}^{k} \frac{i \log p_i}{8} \ge k! 8^{-k} \ge 2^{\Omega(k \log k)}$$

[a] Knowledge of number theory is needed here!

RV with Detection

Rendezvous with $O(1)$ memory

- The main question:

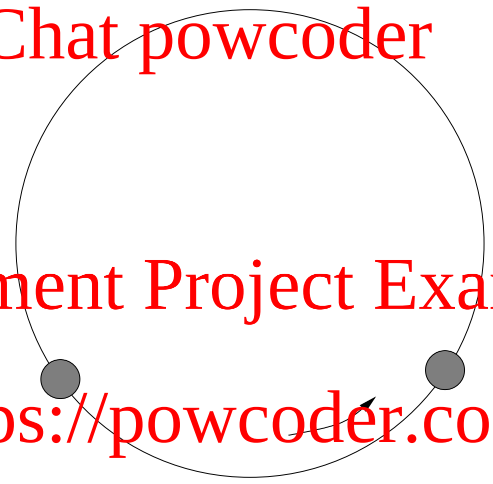
  Can you solve the rendezvous problem if the MAs have constant memory?

- One way or the other you must change the rules of the game!

  – How about if you are allowed to change the position of your token?

Movable Tokens and $O(1)$ memory

n, d, orientation unknown.



1. Release the token
2. Choose a direction and begin walking
3. Upon finding a token, reverse direction.
4. Move the token one node in the new direction
5. Continue walking in the new direction
6. Repeat from step 3 until rendezvous occurs

RVP vs RVD

- **RVP**: denotes the Rendezvous Problem
  - The requirement is to cause rendezvous: if rendezvous is not possible then an algorithm may be running forever without termination, e.g., when the agents are initially placed at distance $n/2$.

Assignment Project Exam Help

- **RVD**: denotes the Rendezvous Problem with detection
  - A rendezvous algorithm is required to detect whether or not rendezvous is possible and if it is to cause rendezvous, otherwise terminate the algorithm.

https://powcoder.com

Add WeChat powcoder

- **NB.** Previous algorithm solves RVP but not RVD.

RVD with Two Movable Tokens

- **Theorem 3** *Rendezvous with detection (RVD) is solvable in a unidirectional ring for two mobile agents with constant memory and two indistinguishable movable tokens each, in time* $O(n^2)$.

- This is based on an algorithm which at the cost of using two tokens per mobile agent detects the possibility of rendezvous and can eventually rendezvous when possible.

RVD with Two Movable Tokens

1. Drop first token at your home base and second token to node located to the right.
2. **repeat**
3. Travel right and move every second token you meet one position to the right.
4. **until** agent detects two tokens on top of each other.
5. **if** two tokens are found on top of each other go around and check if other two tokens are also on top of each other.
6. **if** yes then rendezvous is not possible **else** agent waits at last position.
7. **endif**
8. **endif**

RVD with Two Movable Tokens

- Each mobile agent drops one token to its home node and the other token to the node located to its right.

- Then it travels right and moves every second token one position to the right (note that this will keep the home node tokens at their original locations).

- The process is repeated until the agent detects two tokens on top of each other.

- When this happens, it goes around and checks to see if the other two tokens are also on top of each other.
- If they are, then the home nodes were $n/2$ away, the whole computation was symmetric and the agents can never rendezvous.

RVD with Two Movable Tokens

- If the other tokens are not on top of each other, then the agent waits as the other agent will eventually come to meet it.
- Let us divide the whole computation into rounds of n time steps each – during one round each agent completes a cycle around the ring and both second tokens are moved two steps.
- As the initial distance of the second token from the first token of the next agent is at most $d - 1 \leq n/2 - 1$, the worst case running time of this algorithm is bound by n times the number of rounds plus $n/2$ for the final check, resulting in $n((n/2 - 1)/2) + n/2$, which is in $O(n^2)$.
- This completes the proof of Theorem 3.

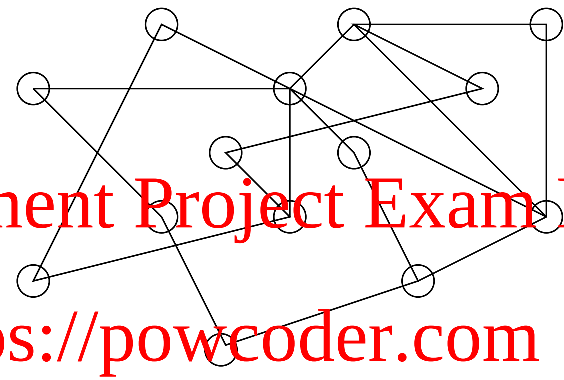
Detecting Rendezvous

Table 1: Time bounds for two mobile agents with constant memory to detect if rendezvous with detection is possible (RVD) and to rendezvous when input is asymmetric (RVP) on an n node synchronous uni-, bi-directional ring with one or two tokens.

Conditions		Time Required for Rendezvous	
# of Tokens	# of Directions	RVD	RVP
1	1	∞	∞
1	2	∞	$\Theta(n^2)$
2	1	$\Theta(n^2)$	$\Theta(n^2)$
2	2	$\Theta(n^2)$	$\Theta(n^2)$

Exercises[a]

1. Give a rendezvous algorithm for two robots at a distinguished node of the graph. What vertex would you choose?



2. Consider a geometric graph (nodes have coordinates). Can you show how to solve the rendezvous problem in a planar geometric graph (geometric means the nodes know their (x, y) coordinates)?

3. Consider a planar graph. Can you show how to solve the rendezvous problem in a planar graph if nodes have no

[a] Do not submit.

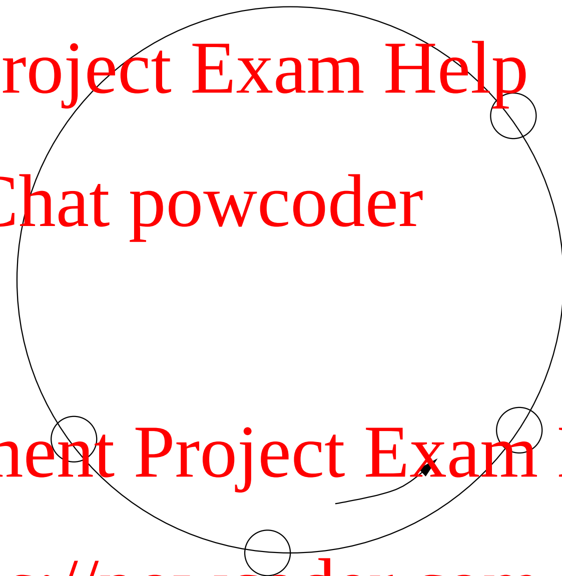
knowledge of their (x, y) coordinates?

4. Prove Theorem 1 by showing that no algorithm exists such that the MAs always correctly detect if $d = \frac{n}{2}$ and act appropriately, i.e., stop if $d = \frac{n}{2}$ and rendezvous otherwise

5. Prove the following for two agents with tokens

Knowledge			Lower	Upper	Memory
n	d	*orientation*	Bound	Bound	Requirement
YES	YES	YES	$3d/4$	$3d/4$	$O(\log d)$
YES	YES	NO	$3n/4$	$5n/6$	$O(\log d)$
YES	NO	YES	$3n/4$	$3n/4$	$O(\log n)$
YES	NO	NO	$3n/4$	$3n/4$	$O(\log n)$
NO	YES	YES	$3n/4$	$3n/4$	$O(\log d)$
NO	YES	NO	$3n/4$	$5n/6$	$O(\log d)$
NO	NO	YES	$5n/4$	$5n/4$	$O(\log n)$
NO	NO	NO	$5n/4$	$5n/4$	$O(\log n)$

6. Consider 4 Mobile Agents (MAs) located on 4 different nodes of an n node ring. They are equipped with indistinguishable tokens which they release at their starting poisions.



its correctness. Let the intertoken distances be d_1, d_2, d_3, d_4. Give a rendezvous algorithm and determine conditions on the initial distances which ensure rendezvous.

References

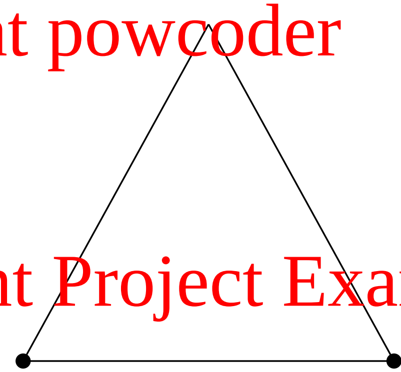
1. S. Alpern and S. Gal, *Theory of Search Games and Rendezvous*, 2003.

2. E. Kranakis, D. Krizanc, and E. Markou, The Mobile Agent Rendezvous Problem in the Ring, Morgan-Claypool, 2010.

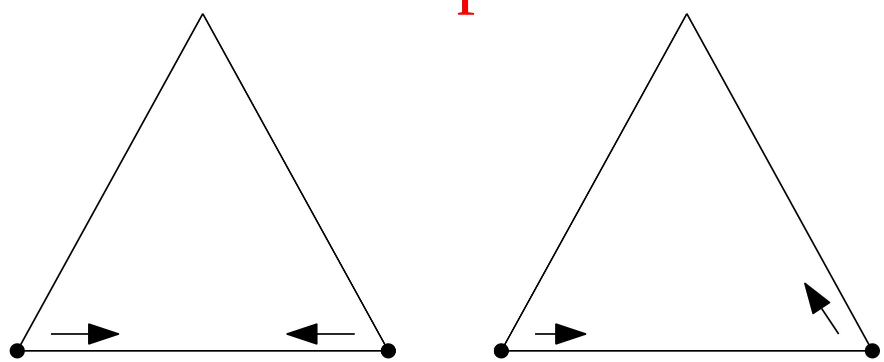
Appendix

Example: Triangle (3-Ring)

- Can two MAs rendezvous in a 3-Ring?

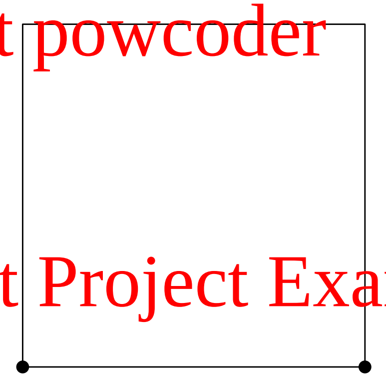


- In lefthand side rendezvous is possibe, but not in the righthand side!

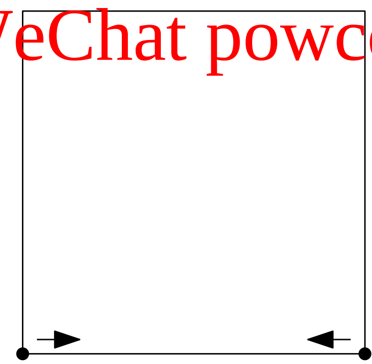


Example: Square (4-Ring)

- Consider two MAs rendezvous in a 4-ring



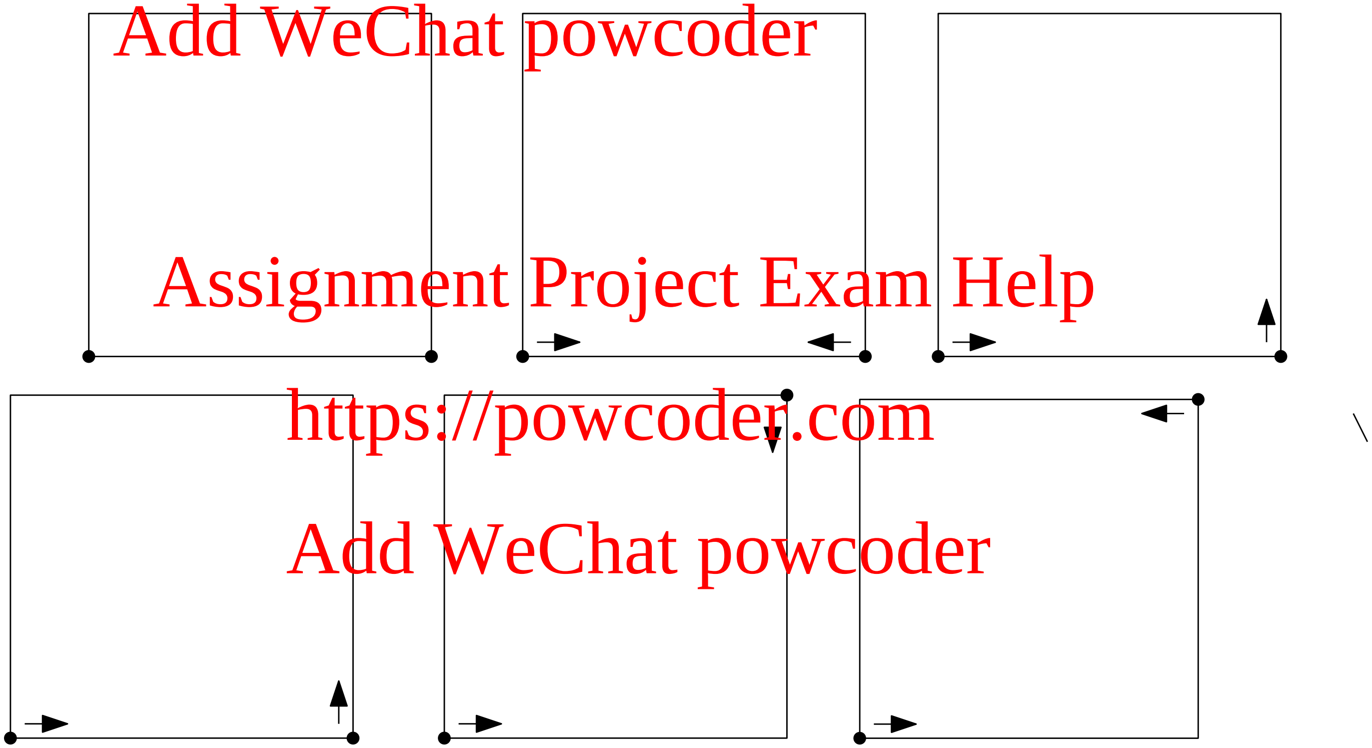
- Rendezvous is never possible on a vertex . . .



. . . but is possible on an edge!

Example: Square (4-Ring)

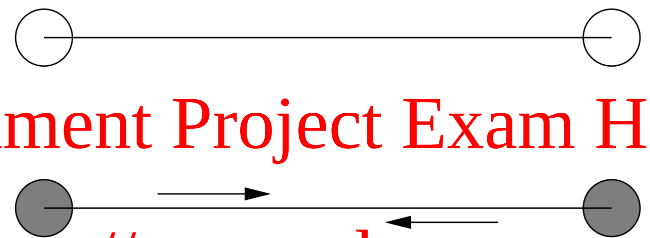
- Consider two MAs



- Algorithm must work in all situations as specified by the given conditions.

A Solution: Rendezvous in an Edge

- If the system is synchronous and the two MAs start at different endpoints of the line they may never be able to rendezvous at a node.



- **Assumption:** rendezvous should be possible to occur either at a node or a link of the network!
  - This simplifies algorithms, otherwise we have to mention the parity of n,

- In general, two different MAs could rendezvous on some node/edge of a network.

The Model: Knowledge

- Knowledge refers to what the agents know about themselves, other agents, and the ring itself.

- Consider k MAs in a ring of n nodes.
  - Do MAs know k (the total number of MAs in the network)?
  - Do MAs know n (the number of nodes in the network)?
  - Do MAs know the topology (orientation)?
  - Do MAs know their initial (inter)-distances?
  - How much memory do the agents need to solve rendezvous?
  - Do the agents have tokens?
- All these characteristics may affect the rendezvous algorithm.

Lonely Runner Conjecture: Wills (1967) and Cusick (1973)

> Suppose k runners having distinct constant speeds start at a common point and run laps on a unit length circular track. Then for any given runner i, there is a time at which runner i is distance at least $1/k$ away from every other runner.

This conjecture was verified for $k \leq 6$ by Cusick and Pomerance (1974) with the assistance of some electronic case checking.

Goddyn, Gvozdjak, Sebo, and Tarsi (1998) supplied an elementary proof of the case $k = 5$, and established a connection between this problem and a question concerning flows on graphs.

Recently Bohman, Holzman, and Kleitman (2001) have solved this problem for $k = 6$ runners.