RUPC 2019 presentation of solutions

RUPC 2019 Jury

- Arnar Bjarni Arnarson
- Bjarki Ágúst Guðmundsson
- Unnar Freyr Erlendsson

A: A Tale of Two Queues

Problem

Two cash registers, and a queue of individuals leading up to each of them. Given how much time each individual will spend at the register, which queue should you enter in order to get to a register as soon as possible?

Solution

A: A Tale of Two Queues

Problem

Two cash registers, and a queue of individuals leading up to each of them. Given how much time each individual will spend at the register, which queue should you enter in order to get to a register as soon as possible?

Solution

1. If you enter a queue, you get to the register at time equal to the sum of times each individual in that queue spends at the register.

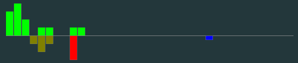
Problem

Two cash registers, and a queue of individuals leading up to each of them. Given how much time each individual will spend at the register, which queue should you enter in order to get to a register as soon as possible?

Solution

1. If you enter a queue, you get to the register at time equal to the sum of times each individual in that queue spends at the register.
2. Let L, R be the sum of times for each individual in the left and right queues, respectively.

A: A Tale of Two Queues



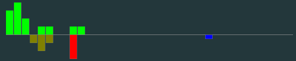
Problem

Two cash registers, and a queue of individuals leading up to each of them. Given how much time each individual will spend at the register, which queue should you enter in order to get to a register as soon as possible?

Solution

1. If you enter a queue, you get to the register at time equal to the sum of times each individual in that queue spends at the register.
2. Let L, R be the sum of times for each individual in the left and right queues, respectively.
3. Three cases:
   - $L < R$: output "`left`"
   - $L > R$: output "`right`"
   - $L = R$: output "`either`"

A: A Tale of Two Queues



Problem

Two cash registers, and a queue of individuals leading up to each of them. Given how much time each individual will spend at the register, which queue should you enter in order to get to a register as soon as possible?

Solution

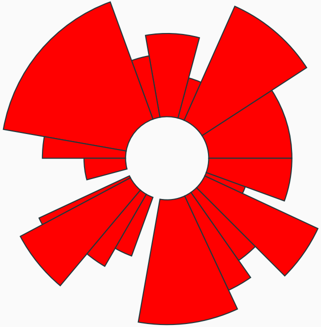
1. If you enter a queue, you get to the register at time equal to the sum of times each individual in that queue spends at the register.
2. Let L, R be the sum of times for each individual in the left and right queues, respectively.
3. Three cases:
   - $L < R$: output "`left`"
   - $L > R$: output "`right`"
   - $L = R$: output "`either`"

Statistics: 20 submissions, 13+ accepted

Problem

Given a description of a figure in terms of disjoint circular sectors, output its area.

Solution



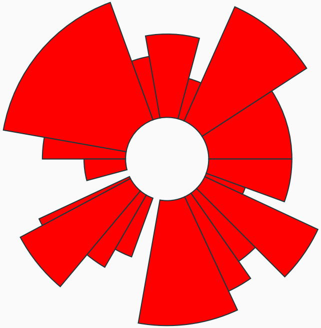
C: Circular Painting

Problem

Given a description of a figure in terms of disjoint circular sectors, output its area.

Solution

1. Circular sectors are non-intersecting.

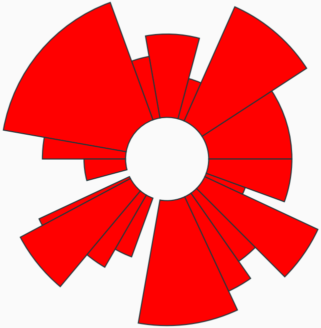


Problem

Given a description of a figure in terms of disjoint circular sectors, output its area.

Solution

1. Circular sectors are non-intersecting.
2. Calculate their areas independently; answer is their sum.



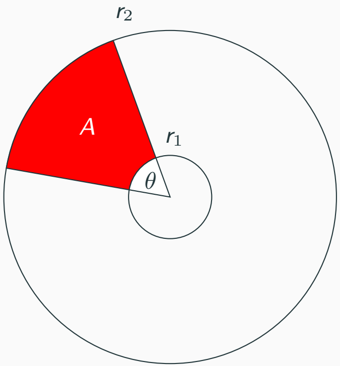
C: Circular Painting

Problem

Given a description of a figure in terms of disjoint circular sectors, output its area.

Solution

1. Circular sectors are non-intersecting.
2. Calculate their areas independently; answer is their sum.

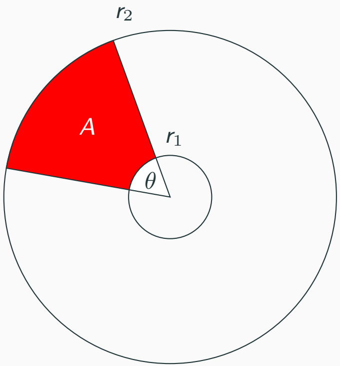


Problem

Given a description of a figure in terms of disjoint circular sectors, output its area.

Solution

1. Circular sectors are non-intersecting.
2. Calculate their areas independently; answer is their sum.
3. $$A = \frac{\theta}{360} \cdot (\pi r_2^2 - \pi r_1^2)$$

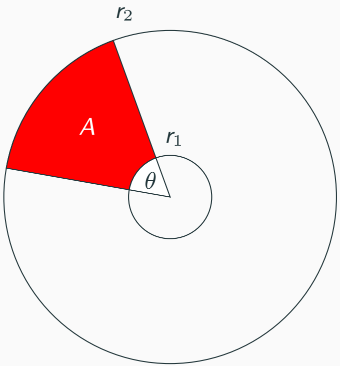


Problem

Given a description of a figure in terms of disjoint circular sectors, output its area.

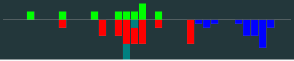
Solution

1. Circular sectors are non-intersecting.
2. Calculate their areas independently; answer is their sum.
3. $$A = \frac{\theta}{360} \cdot \left(\pi r_2^2 - \pi r_1^2\right)$$



Statistics: 11 submissions, 9+ accepted

H: Hot Dog Deals

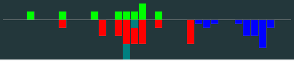


Problem

You can buy one hot dog for 299 ISK, one soda for 249 ISK, one hot dog and one soda for 499 ISK, and two hot dogs and one soda for 549 ISK. What is the minimum amount of ISK needed to get n hot dogs and m sodas?

Solution

H: Hot Dog Deals



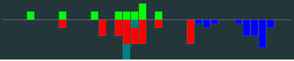
Problem

You can buy one hot dog for 299 ISK, one soda for 249 ISK, one hot dog and one soda for 499 ISK, and two hot dogs and one soda for 549 ISK. What is the minimum amount of ISK needed to get *n* hot dogs and *m* sodas?

Solution

1. Gotcha: May actually be cheaper to buy more than *n* hot dogs or *m* sodas.
   - For two hot dogs, it's better to use the second offer to buy two hot dogs and one soda (549 ISK) than two single hot dogs ($2 \cdot 299 = 598$ ISK).

H: Hot Dog Deals



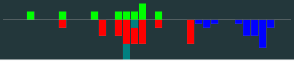
Problem

You can buy one hot dog for 299 ISK, one soda for 249 ISK, one hot dog and one soda for 499 ISK, and two hot dogs and one soda for 549 ISK. What is the minimum amount of ISK needed to get n hot dogs and m sodas?

Solution

1. Gotcha: May actually be cheaper to buy more than n hot dogs or m sodas.
   - For two hot dogs, it's better to use the second offer to buy two hot dogs and one soda (549 ISK) than two single hot dogs ($2 \cdot 299 = 598$ ISK).
2. This observation leads to a greedy algorithm: Use the second offer while we have at least two hot dogs left.

H: Hot Dog Deals



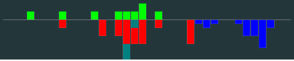
Problem

You can buy one hot dog for 299 ISK, one soda for 249 ISK, one hot dog and one soda for 499 ISK, and two hot dogs and one soda for 549 ISK. What is the minimum amount of ISK needed to get n hot dogs and m sodas?

Solution

1. Greedy algorithm: $c \leftarrow 0$
   - While $n \geq 2$, buy second offer: $c \leftarrow c + 549$; $n \leftarrow n - 2$; $m \leftarrow m - 1$
   - If $n = 1$ and $m \geq 1$, buy first offer: $c \leftarrow c + 499$; $n \leftarrow 0$; $m \leftarrow m - 1$
   - If $n = 1$ and $m = 0$, buy hot dog: $c \leftarrow c + 299$; $n \leftarrow 0$
   - While $m \geq 1$, buy soda: $c \leftarrow c + 249$; $m \leftarrow m - 1$

H: Hot Dog Deals



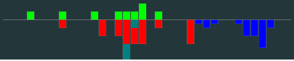
Problem

You can buy one hot dog for 299 ISK, one soda for 249 ISK, one hot dog and one soda for 499 ISK, and two hot dogs and one soda for 549 ISK. What is the minimum amount of ISK needed to get n hot dogs and m sodas?

Solution

1. Greedy algorithm: $c \leftarrow 0$
   - While $n \geq 2$, buy second offer: $c \leftarrow c + 549$; $n \leftarrow n - 2$; $m \leftarrow m - 1$
   - If $n = 1$ and $m \geq 1$, buy first offer: $c \leftarrow c + 499$; $n \leftarrow 0$; $m \leftarrow m - 1$
   - If $n = 1$ and $m = 0$, buy hot dog: $c \leftarrow c + 299$; $n \leftarrow 0$
   - While $m \geq 1$, buy soda: $c \leftarrow c + 249$; $m \leftarrow m - 1$
2. Can be implemented in $O(1)$ time.

H: Hot Dog Deals



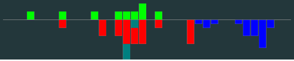
Problem

You can buy one hot dog for 299 ISK, one soda for 249 ISK, one hot dog and one soda for 499 ISK, and two hot dogs and one soda for 549 ISK. What is the minimum amount of ISK needed to get n hot dogs and m sodas?

Solution

1. Greedy algorithm: $c \leftarrow 0$
   - While $n \geq 2$, buy second offer: $c \leftarrow c + 549$; $n \leftarrow n - 2$; $m \leftarrow m - 1$
   - If $n = 1$ and $m \geq 1$, buy first offer: $c \leftarrow c + 499$; $n \leftarrow 0$; $m \leftarrow m - 1$
   - If $n = 1$ and $m = 0$, buy hot dog: $c \leftarrow c + 299$; $n \leftarrow 0$
   - While $m \geq 1$, buy soda: $c \leftarrow c + 249$; $m \leftarrow m - 1$
2. Can be implemented in $O(1)$ time.
3. Other approaches:
   - Loop through number of times we use second offer, and then apply simpler greedy algorithm: $O(n)$
   - Dynamic Programming: $O(nm)$

H: Hot Dog Deals



Problem

You can buy one hot dog for 299 ISK, one soda for 249 ISK, one hot dog and one soda for 499 ISK, and two hot dogs and one soda for 549 ISK. What is the minimum amount of ISK needed to get n hot dogs and m sodas?

Solution

1. Greedy algorithm: $c \leftarrow 0$
   - While $n \geq 2$, buy second offer: $c \leftarrow c + 549$; $n \leftarrow n - 2$; $m \leftarrow m - 1$
   - If $n = 1$ and $m \geq 1$, buy first offer: $c \leftarrow c + 499$; $n \leftarrow 0$; $m \leftarrow m - 1$
   - If $n = 1$ and $m = 0$, buy hot dog: $c \leftarrow c + 299$; $n \leftarrow 0$
   - While $m \geq 1$, buy soda: $c \leftarrow c + 249$; $m \leftarrow m - 1$
2. Can be implemented in $O(1)$ time.
3. Other approaches:
   - Loop through number of times we use second offer, and then apply simpler greedy algorithm: $O(n)$
   - Dynamic Programming: $O(nm)$

Statistics: 27 submissions, 9+ accepted

G: Group Lockers

Problem

There are m columns of lockers, and in column i there are c_i lockers available for rent. What is the smallest span of columns where you can rent n lockers?

Solution

G: Group Lockers

Problem

There are m columns of lockers, and in column i there are c_i lockers available for rent. What is the smallest span of columns where you can rent n lockers?

Solution

1. Task: Find smallest subarray with sum $\geq n$.

G: Group Lockers

Problem

There are m columns of lockers, and in column i there are c_i lockers available for rent. What is the smallest span of columns where you can rent n lockers?

Solution

1. Task: Find smallest subarray with sum $\geq n$.
2. Naive solution: For each left index i, do a linear search for the smallest index j such that $c_i + \cdots + c_j \geq n$.

G: Group Lockers

Problem

There are m columns of lockers, and in column i there are c_i lockers available for rent. What is the smallest span of columns where you can rent n lockers?

Solution

1. Task: Find smallest subarray with sum $\geq n$.
2. Naive solution: For each left index i, do a linear search for the smallest index j such that $c_i + \cdots + c_j \geq n$.

$$n = 8$$

i							
1	0	2	6	3	0	1	7

Problem

There are m columns of lockers, and in column i there are c_i lockers available for rent. What is the smallest span of columns where you can rent n lockers?

Solution

1. Task: Find smallest subarray with sum $\geq n$.
2. Naive solution: For each left index i, do a linear search for the smallest index j such that $c_i + \cdots + c_j \geq n$.

$$n = 8$$

i			j				
1	0	2	6	3	0	1	7

G: Group Lockers

Problem

There are m columns of lockers, and in column i there are c_i lockers available for rent. What is the smallest span of columns where you can rent n lockers?

Solution

1. Task: Find smallest subarray with sum $\geq n$.
2. Naive solution: For each left index i, do a linear search for the smallest index j such that $c_i + \cdots + c_j \geq n$.

$$n = 8$$

	i						
1	0	2	6	3	0	1	7

G: Group Lockers

Problem

There are m columns of lockers, and in column i there are c_i lockers available for rent. What is the smallest span of columns where you can rent n lockers?

Solution

1. Task: Find smallest subarray with sum $\geq n$.
2. Naive solution: For each left index i, do a linear search for the smallest index j such that $c_i + \cdots + c_j \geq n$.

$$n = 8$$

$$\begin{array}{cccccccc} & \text{i} & & \text{j} & & & & \\ 1 & 0 & 2 & 6 & 3 & 0 & 1 & 7 \end{array}$$

G: Group Lockers

Problem

There are m columns of lockers, and in column i there are c_i lockers available for rent. What is the smallest span of columns where you can rent n lockers?

Solution

1. Task: Find smallest subarray with sum $\geq n$.
2. Naive solution: For each left index i, do a linear search for the smallest index j such that $c_i + \cdots + c_j \geq n$.

$$n = 8$$

		i					
1	0	2	6	3	0	1	7

Problem

There are m columns of lockers, and in column i there are c_i lockers available for rent. What is the smallest span of columns where you can rent n lockers?

Solution

1. Task: Find smallest subarray with sum $\geq n$.
2. Naive solution: For each left index i, do a linear search for the smallest index j such that $c_i + \cdots + c_j \geq n$.

$$n = 8$$

		i	j				
1	0	2	6	3	0	1	7

G: Group Lockers

Problem

There are m columns of lockers, and in column i there are c_i lockers available for rent. What is the smallest span of columns where you can rent n lockers?

Solution

1. Task: Find smallest subarray with sum $\geq n$.
2. Naive solution: For each left index i, do a linear search for the smallest index j such that $c_i + \cdots + c_j \geq n$.

$$n = 8$$

			i				
1	0	2	6	3	0	1	7

G: Group Lockers

Problem

There are m columns of lockers, and in column i there are c_i lockers available for rent. What is the smallest span of columns where you can rent n lockers?

Solution

1. Task: Find smallest subarray with sum $\geq n$.
2. Naive solution: For each left index i, do a linear search for the smallest index j such that $c_i + \cdots + c_j \geq n$.

$$n = 8$$

			i	j			
1	0	2	6	3	0	1	7

G: Group Lockers

Problem

There are m columns of lockers, and in column i there are c_i lockers available for rent. What is the smallest span of columns where you can rent n lockers?

Solution

1. Task: Find smallest subarray with sum $\geq n$.
2. Naive solution: For each left index i, do a linear search for the smallest index j such that $c_i + \cdots + c_j \geq n$.

$$n = 8$$

$$\begin{array}{cccccccc} & & & \text{i} & \text{j} & & & \\ 1 & 0 & 2 & \color{red}{6} & \color{red}{3} & 0 & 1 & 7 \end{array}$$

3. Instead of restarting the search for j each time, keep track of the current sum and increment j as needed.

G: Group Lockers

Problem

There are m columns of lockers, and in column i there are c_i lockers available for rent. What is the smallest span of columns where you can rent n lockers?

Solution

$$n = 8$$

$$\text{sum} = 0$$

i							
1	0	2	6	3	0	1	7

G: Group Lockers

Problem

There are m columns of lockers, and in column i there are c_i lockers available for rent. What is the smallest span of columns where you can rent n lockers?

Solution

$$n = 8$$

$$\text{sum} = 9$$

i			j				
1	0	2	6	3	0	1	7

Problem

There are m columns of lockers, and in column i there are c_i lockers available for rent. What is the smallest span of columns where you can rent n lockers?

Solution

$$n = 8$$

$$\text{sum} = 8$$

$$\begin{array}{cccccccc} & i & & j & & & & \\ 1 & \color{red}{0} & \color{red}{2} & \color{red}{6} & 3 & 0 & 1 & 7 \end{array}$$

G: Group Lockers

Problem

There are m columns of lockers, and in column i there are c_i lockers available for rent. What is the smallest span of columns where you can rent n lockers?

Solution

$$n = 8$$

$$\text{sum} = 8$$

		i	j				
1	0	2	6	3	0	1	7

Problem

There are m columns of lockers, and in column i there are c_i lockers available for rent. What is the smallest span of columns where you can rent n lockers?

Solution

$$n = 8$$

$$\text{sum} = 6$$

			ij				
1	0	2	6	3	0	1	7

G: Group Lockers

Problem

There are m columns of lockers, and in column i there are c_i lockers available for rent. What is the smallest span of columns where you can rent n lockers?

Solution

$$n = 8$$

$$\text{sum} = 9$$

$$\begin{array}{cccccccc} & & & \text{i} & \text{j} & & & \\ 1 & 0 & 2 & \color{red}{6} & \color{red}{3} & 0 & 1 & 7 \end{array}$$

G: Group Lockers

Problem

There are m columns of lockers, and in column i there are c_i lockers available for rent. What is the smallest span of columns where you can rent n lockers?

Solution

$$n = 8$$

$$\text{sum} = 9$$

$$\begin{array}{cccccccc} & & & i & j & & & \\ 1 & 0 & 2 & \color{red}{6} & \color{red}{3} & 0 & 1 & 7 \end{array}$$

1. i is incremented at most n times, j is incremented at most n times: $O(n)$

G: Group Lockers

Problem

There are m columns of lockers, and in column i there are c_i lockers available for rent. What is the smallest span of columns where you can rent n lockers?

Solution

$$n = 8$$

$$\text{sum} = 9$$

$$\begin{array}{cccccccc} & & & \text{i} & \text{j} & & & \\ 1 & 0 & 2 & \color{red}{6} & \color{red}{3} & 0 & 1 & 7 \end{array}$$

1. i is incremented at most n times, j is incremented at most n times: $O(n)$
2. Other approaches:
   - Compute prefix sums, and then binary search. $O(n \log n)$

G: Group Lockers

Problem

There are m columns of lockers, and in column i there are c_i lockers available for rent. What is the smallest span of columns where you can rent n lockers?

Solution

$$n = 8$$

$$\text{sum} = 9$$

			i	j			
1	0	2	6	3	0	1	7

1. i is incremented at most n times, j is incremented at most n times: $O(n)$
2. Other approaches:
   - Compute prefix sums, and then binary search. $O(n \log n)$

Statistics: 13 submissions, 3+ accepted

B: Back to Studying

Problem

There are n final exams. You need to spend c_i days studying for ith exam, which will be held on day d_i. How many days until you must start studying?

Solution

B: Back to Studying

Problem

There are n final exams. You need to spend c_i days studying for ith exam, which will be held on day d_i. How many days until you must start studying?

Solution

1. For each exam, we want to start studying as late as possible; ideally on day $d_i - c_i$.

B: Back to Studying

Problem

There are n final exams. You need to spend c_i days studying for ith exam, which will be held on day d_i. How many days until you must start studying?

Solution

1. For each exam, we want to start studying as late as possible; ideally on day $d_i - c_i$.
2. However, we may already be studying for a later exam during some of those days. In that case we can just start our preparations for exam i a couple days earlier.

B: Back to Studying

Problem

There are n final exams. You need to spend c_i days studying for ith exam, which will be held on day d_i. How many days until you must start studying?

Solution

1. For each exam, we want to start studying as late as possible; ideally on day $d_i - c_i$.
2. However, we may already be studying for a later exam during some of those days. In that case we can just start our preparations for exam i a couple days earlier.
3. Going through the exams in decreasing order of d_i, we can keep track of how many more days of studying we need for the exams seen so far, and decrease this counter as we move past days where we can study.

B: Back to Studying

Problem

There are n final exams. You need to spend c_i days studying for ith exam, which will be held on day d_i. How many days until you must start studying?

Solution

1. For each exam, we want to start studying as late as possible; ideally on day $d_i - c_i$.
2. However, we may already be studying for a later exam during some of those days. In that case we can just start our preparations for exam i a couple days earlier.
3. Going through the exams in decreasing order of d_i, we can keep track of how many more days of studying we need for the exams seen so far, and decrease this counter as we move past days where we can study.
4. If we get to day 0 and still have more than one day of studying left, output "`impossible`". Otherwise, output the earliest day we used for studying.

B: Back to Studying

Problem

There are n final exams. You need to spend c_i days studying for ith exam, which will be held on day d_i. How many days until you must start studying?

Solution

1. For each exam, we want to start studying as late as possible; ideally on day $d_i - c_i$.
2. However, we may already be studying for a later exam during some of those days. In that case we can just start our preparations for exam i a couple days earlier.
3. Going through the exams in decreasing order of d_i, we can keep track of how many more days of studying we need for the exams seen so far, and decrease this counter as we move past days where we can study.
4. If we get to day 0 and still have more than one day of studying left, output "`impossible`". Otherwise, output the earliest day we used for studying.

Statistics: 21 submissions, 2+ accepted

E: Emergency Exits

Problem

Given a directed, weighted graph, and a set of vertices that are emergency exits, find the vertex that is farthest away from all emergency exits.

Solution

Problem

Given a directed, weighted graph, and a set of vertices that are emergency exits, find the vertex that is farthest away from all emergency exits.

Solution

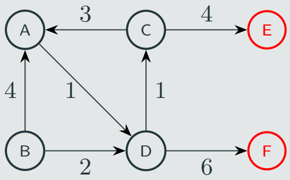
1. Slow solution: For each vertex find shortest paths to all other vertices using Dijkstra's algorithm.

Problem

Given a directed, weighted graph, and a set of vertices that are emergency exits, find the vertex that is farthest away from all emergency exits.

Solution

1. Slow solution: For each vertex find shortest paths to all other vertices using Dijkstra's algorithm.



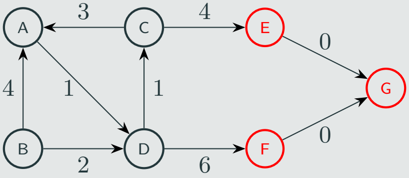
E: Emergency Exits

Problem

Given a directed, weighted graph, and a set of vertices that are emergency exits, find the vertex that is farthest away from all emergency exits.

Solution

1. Slow solution: For each vertex find shortest paths to all other vertices using Dijkstra's algorithm.



2. Trick: We can add a special node to represent all emergency exits.

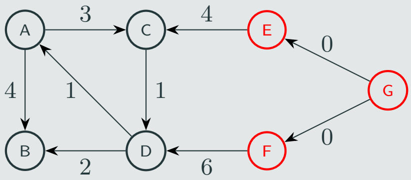
E: Emergency Exits

Problem

Given a directed, weighted graph, and a set of vertices that are emergency exits, find the vertex that is farthest away from all emergency exits.

Solution

1. Slow solution: For each vertex find shortest paths to all other vertices using Dijkstra's algorithm.



2. Trick: We can add a special node to represent all emergency exits.
3. Trick: We can reverse the edges. Now we just need to run Dijkstra once from G: $O(m \log n)$

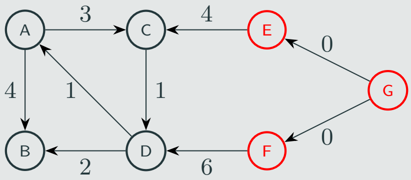
E: Emergency Exits

Problem

Given a directed, weighted graph, and a set of vertices that are emergency exits, find the vertex that is farthest away from all emergency exits.

Solution

1. Slow solution: For each vertex find shortest paths to all other vertices using Dijkstra's algorithm.



2. Trick: We can add a special node to represent all emergency exits.
3. Trick: We can reverse the edges. Now we just need to run Dijkstra once from G: $O(m \log n)$

Statistics: 1 submissions, 1+ accepted

F: Fetch the Stuff

Problem

There are a set of doors between rooms in a building. Some doors can only be used in one direction. Given a set of important rooms, find a walk around the building that goes from room 1 to room n, and visits all important rooms.

Solution

Problem

There are a set of doors between rooms in a building. Some doors can only be used in one direction. Given a set of important rooms, find a walk around the building that goes from room 1 to room n, and visits all important rooms.

Solution

1. Can be viewed as a directed graph: rooms are vertices and doors are edges from/to/between different rooms.

F: Fetch the Stuff

Problem

There are a set of doors between rooms in a building. Some doors can only be used in one direction. Given a set of important rooms, find a walk around the building that goes from room 1 to room n, and visits all important rooms.

Solution

1. Can be viewed as a directed graph: rooms are vertices and doors are edges from/to/between different rooms.
2. Necessary condition: Room 1 must be able to reach all important rooms, and all important rooms must be able to reach room n.

Problem

There are a set of doors between rooms in a building. Some doors can only be used in one direction. Given a set of important rooms, find a walk around the building that goes from room 1 to room n, and visits all important rooms.

Solution

1. Can be viewed as a directed graph: rooms are vertices and doors are edges from/to/between different rooms.
2. Necessary condition: Room 1 must be able to reach all important rooms, and all important rooms must be able to reach room n.
3. Now consider two important rooms u and v. If there is no path from u to v, then we must visit room v before room u.

F: Fetch the Stuff

Problem

There are a set of doors between rooms in a building. Some doors can only be used in one direction. Given a set of important rooms, find a walk around the building that goes from room 1 to room n, and visits all important rooms.

Solution

1. Can be viewed as a directed graph: rooms are vertices and doors are edges from/to/between different rooms.
2. Necessary condition: Room 1 must be able to reach all important rooms, and all important rooms must be able to reach room n.
3. Now consider two important rooms u and v. If there is no path from u to v, then we must visit room v before room u.
4. This can be turned into a "dependency" graph, where there is an edge from vertex v to vertex u if v must be visited before vertex u.

F: Fetch the Stuff

Problem

There are a set of doors between rooms in a building. Some doors can only be used in one direction. Given a set of important rooms, find a walk around the building that goes from room 1 to room n, and visits all important rooms.

Solution (ctd)

5. What we're now looking for is a "topological order" of the rooms in this new graph. Can be found with DFS, or determined that there is no such order.

F: Fetch the Stuff

Problem

There are a set of doors between rooms in a building. Some doors can only be used in one direction. Given a set of important rooms, find a walk around the building that goes from room 1 to room n, and visits all important rooms.

Solution (ctd)

5. What we're now looking for is a "topological order" of the rooms in this new graph. Can be found with DFS, or determined that there is no such order.
6. To construct the walk, go through the important rooms in a topological order, and find paths between them with DFS.

F: Fetch the Stuff

Problem

There are a set of doors between rooms in a building. Some doors can only be used in one direction. Given a set of important rooms, find a walk around the building that goes from room 1 to room n, and visits all important rooms.

Solution (ctd)

5. What we're now looking for is a "topological order" of the rooms in this new graph. Can be found with DFS, or determined that there is no such order.
6. To construct the walk, go through the important rooms in a topological order, and find paths between them with DFS.
7. DFS from important rooms to all other rooms, topological sort, and reconstruct answer: $O(kn + k + kn) = O(kn)$

Problem

There are a set of doors between rooms in a building. Some doors can only be used in one direction. Given a set of important rooms, find a walk around the building that goes from room 1 to room n, and visits all important rooms.

Solution (ctd)

5. What we're now looking for is a "topological order" of the rooms in this new graph. Can be found with DFS, or determined that there is no such order.
6. To construct the walk, go through the important rooms in a topological order, and find paths between them with DFS.
7. DFS from important rooms to all other rooms, topological sort, and reconstruct answer: $O(kn + k + kn) = O(kn)$
8. Other approaches:
   - Dynamic Programming over Strongly Connected Components: $O(nk)$
   - Backtracking with some clever pruning: $O(nk)$

F: Fetch the Stuff

Problem

There are a set of doors between rooms in a building. Some doors can only be used in one direction. Given a set of important rooms, find a walk around the building that goes from room 1 to room n, and visits all important rooms.

Solution (ctd)

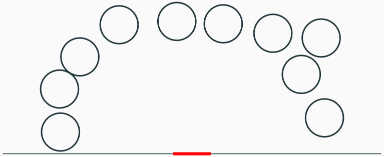
5. What we're now looking for is a "topological order" of the rooms in this new graph. Can be found with DFS, or determined that there is no such order.
6. To construct the walk, go through the important rooms in a topological order, and find paths between them with DFS.
7. DFS from important rooms to all other rooms, topological sort, and reconstruct answer: $O(kn + k + kn) = O(kn)$
8. Other approaches:
   - Dynamic Programming over Strongly Connected Components: $O(nk)$
   - Backtracking with some clever pruning: $O(nk)$

Statistics: 2 submissions, 1+ accepted

D: Danger Zone

Problem

A group of n circles above an infinite line, with an entrance in the middle. Starting from far away, is it possible to reach the entrance without coming too close to the wall or the circles?

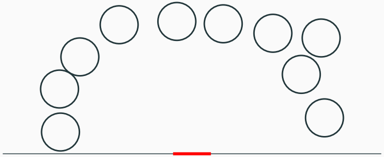


Solution

D: Danger Zone

Problem

A group of n circles above an infinite line, with an entrance in the middle. Starting from far away, is it possible to reach the entrance without coming too close to the wall or the circles?



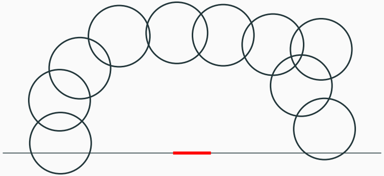
Solution

1. To deal with the minimum distance restriction, we can increase the radius of each circle, and move the infinite line.

D: Danger Zone

Problem

A group of n circles above an infinite line, with an entrance in the middle. Starting from far away, is it possible to reach the entrance without coming too close to the wall or the circles?



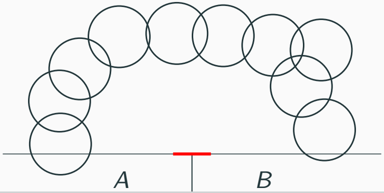
Solution

1. To deal with the minimum distance restriction, we can increase the radius of each circle, and move the infinite line.

D: Danger Zone

Problem

A group of n circles above an infinite line, with an entrance in the middle. Starting from far away, is it possible to reach the entrance without coming too close to the wall or the circles?



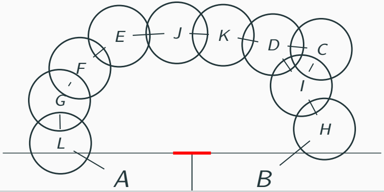
Solution

1. To deal with the minimum distance restriction, we can increase the radius of each circle, and move the infinite line.
2. Split the store into two parts.

D: Danger Zone

Problem

A group of n circles above an infinite line, with an entrance in the middle. Starting from far away, is it possible to reach the entrance without coming too close to the wall or the circles?



Solution

1. To deal with the minimum distance restriction, we can increase the radius of each circle, and move the infinite line.
2. Split the store into two parts.
3. Create a graph with circles and A and B as vertices. Edge between two objects if they intersect.

Solution (ctd.)

4. If there is a path from A to B, the entrance is sealed.

Solution (ctd.)

4. If there is a path from A to B, the entrance is sealed.
5. Otherwise there is always a way in.

Solution (ctd.)

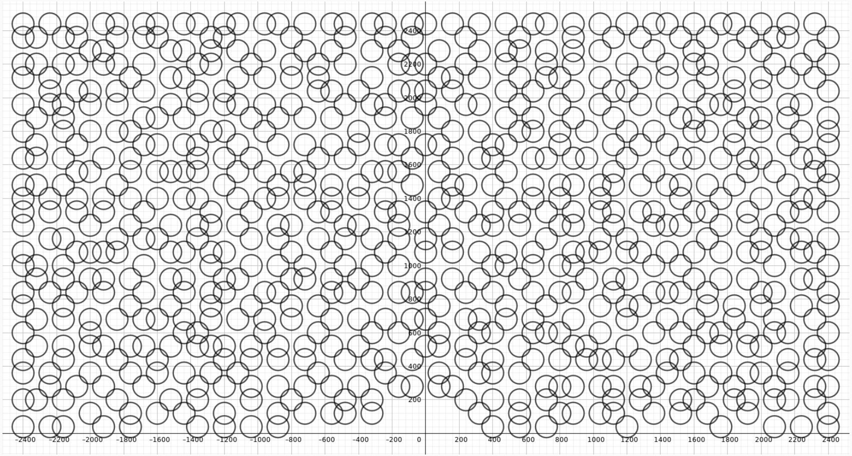
4. If there is a path from A to B, the entrance is sealed.
5. Otherwise there is always a way in.
6. Use BFS/DFS/Union-Find: $O(n^2)$

D: Danger Zone

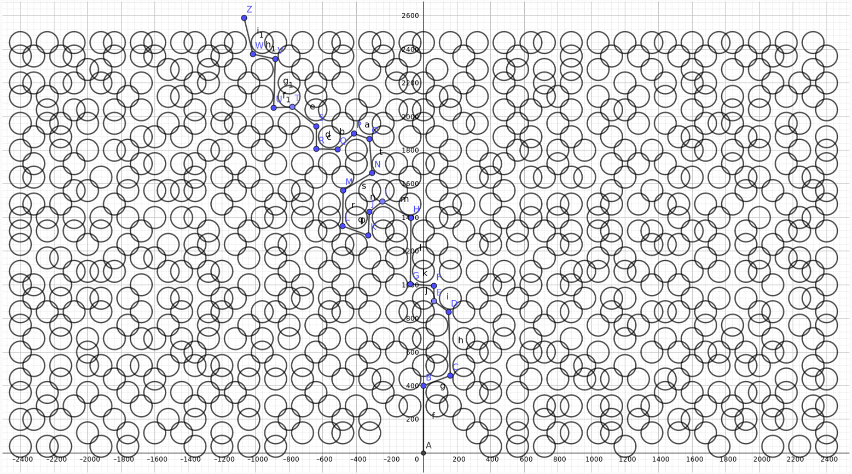
Solution (ctd.)

4. If there is a path from A to B, the entrance is sealed.
5. Otherwise there is always a way in.
6. Use BFS/DFS/Union-Find: $O(n^2)$
7. Use integers for all distance comparisons.

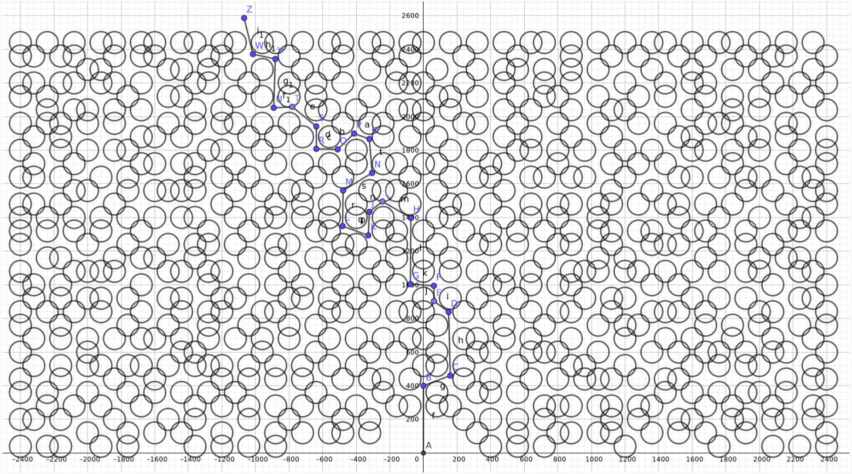
D: Danger Zone



D: Danger Zone



D: Danger Zone



Statistics: 6 submissions, 0+ accepted