Dancing Mice

Petr Mitrichev

SOI Day, January 8, 2022

Task description

N mice are dancing in a circle, at each moment in time each mouse may dance together with at most one of its neighbors. You need to design a dance with the shortest overall duration that allows mice i and $i+1$ to dance as a pair for at least t_i seconds for each i.

Example

Input:

```
5
1 1 1 1 1
```

Output:

```
5
0.5 <><>.
0.5 <>.<>
0.5 .<><>
0.5 ><>.<
0.5 >.<><
```

Solution

In subtask 1 worth 5 points, there are exactly three mice: $N = 3$.

Solution

In subtask 1 worth 5 points, there are exactly three mice: $N = 3$.

- Only one pair of mice can dance together at the same time, so we just need to stack together the three segments, one for each pair of mice.
- The duration of the resulting dance is simply the sum of the input durations.

Solution

In subtask 2 worth 10 points, all dance durations are equal to 1: $t_i = 1$.

Solution

In subtask 2 worth 10 points, all dance durations are equal to 1: $t_i = 1$.

- There were two sample cases showing the answers for $N = 4$ and $N = 5$, which we can generalize from.

Solution

In subtask 2 worth 10 points, all dance durations are equal to 1: $t_i = 1$.

- There were two sample cases showing the answers for $N = 4$ and $N = 5$, which we can generalize from.
- For even N, just two dance segments of length 1: odd pairs and even pairs.

Solution

In subtask 2 worth 10 points, all dance durations are equal to 1: $t_i = 1$.

- There were two sample cases showing the answers for $N = 4$ and $N = 5$, which we can generalize from.
- For even N, just two dance segments of length 1: odd pairs and even pairs.
- For odd N, have N segments of length $\frac{2}{N-1}$: in the i-th segment all mice except mouse i are dancing.

Solution

In subtask 2 worth 10 points, all dance durations are equal to 1: $t_i = 1$.

- There were two sample cases showing the answers for $N = 4$ and $N = 5$, which we can generalize from.
- For even N, just two dance segments of length 1: odd pairs and even pairs.
- For odd N, have N segments of length $\frac{2}{N-1}$: in the i-th segment all mice except mouse i are dancing.
- This is optimal since we always have the maximum number of dancing mice, and reach exactly the required dance durations in the end.

Approach

In subtask 3 worth 12 points, $t_{N-1} = 0$, so effectively we have a chain instead of a circle.

Approach

In subtask 3 worth 12 points, $t_{N-1} = 0$, so effectively we have a chain instead of a circle.

- $max_i(t_i + t_{i+1})$ is a lower boundary on the answer, since this is how much mouse $i + 1$ needs to dance. Let us define $s_i = t_i + t_{i+1}$.

Approach

In subtask 3 worth 12 points, $t_{N-1} = 0$, so effectively we have a chain instead of a circle.

- $max_i(t_i + t_{i+1})$ is a lower boundary on the answer, since this is how much mouse $i + 1$ needs to dance. Let us define $s_i = t_i + t_{i+1}$.
- If we can always find a way to dance 1 second that decreases $max_i s_i$ by 1, then this is not just lower boundary but exact answer!

Example

$t =$	1	2	3	2	0
$s =$	3	5	5	2	1

Example

$t =$	1	2	3	2	0
$s =$	3	5	5	2	1

- Suppose the pairs highlighted with gray (t_0 and t_2) are dancing. Then four values of s_i (also highlighted with gray) are decreasing.

Example

$t =$	1	2	3	2	0
$s =$	3	5	5	2	1

- Suppose the pairs highlighted with gray (t_0 and t_2) are dancing. Then four values of s_i (also highlighted with gray) are decreasing.
- The only non-decreasing s_i is $s_3 = 2 + 0$, but it's not the maximum, so $max_i s_i$ will still decrease.

Example

$t =$	1	2	3	2	0
$s =$	3	5	5	2	1

- Suppose the pairs highlighted with gray (t_0 and t_2) are dancing. Then four values of s_i (also highlighted with gray) are decreasing.
- The only non-decreasing s_i is $s_3 = 2 + 0$, but it's not the maximum, so $max_i s_i$ will still decrease.
- We can keep decreasing until one of the values we decrease reaches 0.

Solution

Let us generalize the approach from the example: for each segment of consecutive non-zero t_i, let us take the leftmost, 3rd leftmost, and so on.

Solution

Let us generalize the approach from the example: for each segment of consecutive non-zero t_i, let us take the leftmost, 3rd leftmost, and so on.

- Having chosen the set of dancing pairs, let them dance until one of the t_i reaches 0.

Solution

Let us generalize the approach from the example: for each segment of consecutive non-zero t_i, let us take the leftmost, 3rd leftmost, and so on.

- Having chosen the set of dancing pairs, let them dance until one of the t_i reaches 0.
- This way we need at most $N - 1$ iterations until we solve the problem, and the total duration will be $max_i s_i$.

Solution

Let us generalize the approach from the example: for each segment of consecutive non-zero t_i, let us take the leftmost, 3rd leftmost, and so on.

- Having chosen the set of dancing pairs, let them dance until one of the t_i reaches 0.
- This way we need at most $N - 1$ iterations until we solve the problem, and the total duration will be $max_i s_i$.
- There are many alternative approaches that work as well here.

Solution

In subtask 4 worth 16 points, N is even.

Solution

In subtask 4 worth 16 points, N is even.

- If there is some $t_i = 0$, we can just use the solution from previous subtask.

Solution

In subtask 4 worth 16 points, N is even.

- If there is some $t_i = 0$, we can just use the solution from previous subtask.
- And if all $t_i > 0$, then we can decrease all s_i, and therefore also $max_i s_i$ by simply letting all even-numbered pairs dance.

Solution

In subtask 4 worth 16 points, N is even.

- If there is some $t_i = 0$, we can just use the solution from previous subtask.
- And if all $t_i > 0$, then we can decrease all s_i, and therefore also $max_i s_i$ by simply letting all even-numbered pairs dance.
- They can keep dancing until we reach the state where $t_i = 0$ for some i, and then switch to the solution from the previous subtask.

Solution

In subtask 4 worth 16 points, N is even.

- If there is some $t_i = 0$, we can just use the solution from previous subtask.
- And if all $t_i > 0$, then we can decrease all s_i, and therefore also $max_i s_i$ by simply letting all even-numbered pairs dance.
- They can keep dancing until we reach the state where $t_i = 0$ for some i, and then switch to the solution from the previous subtask.
- The total duration will still be $max_i s_i$, and therefore this is optimal.

Approach

In subtask 5 worth 57 points, N is odd.

Approach

In subtask 5 worth 57 points, N is odd.

- It is no longer true that the answer is $max_i s_i$, we have seen counter-examples in subtasks 1 and 2.

Approach

In subtask 5 worth 57 points, N is odd.

- It is no longer true that the answer is $max_i s_i$, we have seen counter-examples in subtasks 1 and 2.
- But those subtasks taught us a different lower boundary: $\frac{2 \cdot \sum_i t_i}{N-1}$, because at most $\frac{N-1}{2}$ pairs can be dancing at the same time.

Approach

In subtask 5 worth 57 points, N is odd.

- It is no longer true that the answer is $max_i s_i$, we have seen counter-examples in subtasks 1 and 2.
- But those subtasks taught us a different lower boundary: $\frac{2 \cdot \sum_i t_i}{N-1}$, because at most $\frac{N-1}{2}$ pairs can be dancing at the same time.
- Is $\max(\max_i s_i, \frac{2 \cdot \sum_i t_i}{N-1})$ always the answer?

Approach

In subtask 5 worth 57 points, N is odd.

- It is no longer true that the answer is $max_i s_i$, we have seen counter-examples in subtasks 1 and 2.
- But those subtasks taught us a different lower boundary: $\frac{2 \cdot \sum_i t_i}{N-1}$, because at most $\frac{N-1}{2}$ pairs can be dancing at the same time.
- Is $\max(\max_i s_i, \frac{2 \cdot \sum_i t_i}{N-1})$ always the answer?
- To prove or disprove, let us try to find a dance segment that will decrease this value.

Solution

Decreasing $\max(\max_i s_i, \frac{2 \cdot \sum_i t_i}{N-1})$:

- We decrease $\frac{2 \cdot \sum_i t_i}{N-1}$ if and only if $\frac{N-1}{2}$ pairs are dancing, which means that exactly one mouse is not dancing.

Solution

Decreasing $\max(\max_i s_i, \frac{2 \cdot \sum_i t_i}{N-1})$:

- We decrease $\frac{2 \cdot \sum_i t_i}{N-1}$ if and only if $\frac{N-1}{2}$ pairs are dancing, which means that exactly one mouse is not dancing.
- If mouse j is not dancing, then all s_i are decreasing except s_{j-1}.

Solution

Decreasing $\max(\max_i s_i, \frac{2 \cdot \sum_i t_i}{N-1})$:

- We decrease $\frac{2 \cdot \sum_i t_i}{N-1}$ if and only if $\frac{N-1}{2}$ pairs are dancing, which means that exactly one mouse is not dancing.
- If mouse j is not dancing, then all s_i are decreasing except s_{j-1}.
- So we just need to pick such j that s_{j-1} is not the maximum, and we have found a way to decrease $\max_i s_i$.

Solution

Decreasing $\max(\max_i s_i, \frac{2 \cdot \sum_i t_i}{N-1})$:

- We decrease $\frac{2 \cdot \sum_i t_i}{N-1}$ if and only if $\frac{N-1}{2}$ pairs are dancing, which means that exactly one mouse is not dancing.
- If mouse j is not dancing, then all s_i are decreasing except s_{j-1}.
- So we just need to pick such j that s_{j-1} is not the maximum, and we have found a way to decrease $\max_i s_i$.
- If there is no such j, it means all s_i are equal, which for odd N means that all t_i are equal, so we can just use the solution from subtask 2!

Solution

Decreasing $\max(\max_i s_i, \frac{2 \cdot \sum_i t_i}{N-1})$:

- We decrease $\frac{2 \cdot \sum_i t_i}{N-1}$ if and only if $\frac{N-1}{2}$ pairs are dancing, which means that exactly one mouse is not dancing.
- If mouse j is not dancing, then all s_i are decreasing except s_{j-1}.
- So we just need to pick such j that s_{j-1} is not the maximum, and we have found a way to decrease $\max_i s_i$.
- If there is no such j, it means all s_i are equal, which for odd N means that all t_i are equal, so we can just use the solution from subtask 2!
- We pick the duration of the dance that either makes $t_i = 0$, in which case we can switch to subtask 3, or makes s_{j-1} equal to $\max_i s_i$.

Final thoughts

- In each iteration, the number of s_i equal to $\max_i s_i$ increases, so we will reach subtask 2 or subtask 3 after at most $N - 1$ iterations, therefore our solution needs at most $2 \cdot N - 1$ steps in total.
- Learned this approach by looking at the code by Vivienne Burckhardt and Ursus Wigger. There are many approaches that work, but this one seems to be the simplest.
- Overall concept of looking for lower boundaries when we need to minimize is called *duality*, and appears everywhere in algorithms and mathematics.
- Thanks for listening!