2024 UDPC

Official Solutions

UDPC 2024 - Writer / Tester

- ✓ Writer
  - 99asdfg
  - mskim17
  - bnb2011
  - ekwoo
  - leo020630
  - jbh0224
  - andrew1010
  - queued_q
- ✓ Tester
  - hyperbolic
  - lky7674
  - mjhmjh1104
  - nick832
  - slah007
  - tony9402
  - utilforever

Junior Division	Problem	Difficulty Intention	Writer
A	Raid the Pink Bean	**Easy**	jbh0224
B	Good Luck Have Fun	**Easy**	leo020630
C	Reserved Seat 1	**Easy**	99asdfg
D	Double Up	**Easy**	bnb2011
E	UDP Stack	**Medium**	leo020630
F	Hyundai Mobis Autonomous Driving Testing 1	**Medium**	leo020630
G	Twitch Plays Pokemon	**Medium**	queued_q
H	Landscape Painting	**Hard**	leo020630
I	Course Registration	**Hard**	99asdfg

Senior Division	Problem	Difficulty Intention	Writer
A	Raid the Pink Bean	**Easy**	jbh0224
B	Good Luck Have Fun	**Easy**	leo020630
C	Double Up	**Easy**	bnb2011
D	Hyundai Mobis Autonomous Driving Testing 2	**Medium**	leo020630
E	Reserved Seat 2	**Medium**	99asdfg
F	Moving Walkway	**Hard**	leo020630
G	Course Registration	**Hard**	99asdfg
H	Hyper Tree Problem	**Hard**	bnb2011
I	Ponix and Dalgoo	**Hard**	bnb2011

1A/2A. Raid the Pink Bean

`implementation`

Difficulty Intention – **Easy**

- Statistics:
  - Junior: 29 Submissions, 17 Accepted, AC rates - 58.621%
  - Senior: 46 Submissions, 28 Accepted, AC rates - 60.870%
- First Solve:
  - Junior: **김재환**POSTECH, 4 min.
  - Senior: **김경민**POSTECH, 2 min.
- Writer: jbh0224

1A/2A. Raid the Pink Bean

- ✓ Since the time interval of each mascot's skill is less than or equal to the damage it deals, the damage per second that each mascot is greater than or equal to 1.
- ✓ Therefore, the maximum time to defeat Pink Bin is approximately $5000/3$, so we can solve it using a loop statement.
- ✓ Increasing the time by 1 second at a time, the mascot that needs to use a skill at the current moment, whether Yunee, Dalgoo, or Ponix, decreases Pink Bin's health by using their skill.
- ✓ Whether a skill should be used at the current moment can be determined by checking if the remainder of the current time divided by the period C is 0.
- ✓ When Pink Bin's health be **less than or equal to** 0, print the current time.

1B/2B. Good Luck Have Fun

`bruteforcing`

Difficulty Intention – **Easy**

- ✓ Statistics:
  - Junior: 38 Submissions, 14 Accepted, AC rates - 36.842%
  - Senior: 58 Submissions, 23 Accepted, AC rates - 39.655%
- ✓ First Solve:
  - Junior: **김재환**[POSTECH], 11 min.
  - Senior: **이유담**[POSTECH], 8 min.
- ✓ Writer: Leo Jang[`leo020630`]

1B/2B. Good Luck Have Fun

- Let's fix the number of stems to be used as X. X is an integer ranging from A to 1000 inclusive.
- For the fixed X, the number of additional leaves that need to be used, denoted as $f(X)$, is as follows:
  - if $3X \leq B \leq 4X$: $f(X) = 0$
  - if $B < 3X$: $f(X) = 3X - B$
- We do not consider the case where $4X < B$ as it is not possible.
- Therefore, for all $A \leq X \leq 1000$, the answer is the minimum value of the sum of the number of stems and leaves that need to be used, $\min(X - A + f(X))$.
- The time complexity is $\mathcal{O}(1000T)$.
- Furthermore, with some case works, each case can be solved $\mathcal{O}(1)$.

2C. Reserved Seat 1

`greedy, sorting`
Difficulty Intention – **Easy**

- ✓ Statistics:
  - Junior: 31 Submissions, 17 Accepted, AC rates - 54.839%
- ✓ First Solve:
  - Junior: **김재환**[POSTECH], 21 min.
- ✓ Writer: 김도훈[99asdfg]

2C. Reserved Seat 1

- ✓ As every height in problem are calculated by (Seat height + Student's height), we can assume that every student's height is $D \times i$ more taller.
- ✓ After increasing every students' height by $D \times i$, Let $H_{i,j}$ be the height of j-th shortest student in i-th row.
- ✓ In this case, $H_{1,1} < H_{2,1} < \cdots < H_{N,1}$ must be satisfied to assign every students while satisfying the condition.
  - If $H_{i,1} \geq H_{i+1,1}$ is true, then every students in i-th row is taller than or equal to $H_{i+1,1}$ by the definition of $H_{i,j}$, which leads the student $H_{i+1,1}$ can't be assigned while satisfying the condition.

- ✓ Furthermore, $H_{1,2} < H_{2,2} < \cdots < H_{N,2}$ must be satisfied, too.
  - If $H_{i,2} \geq H_{i+1,2}$ is true, every students in i-th row excluding $H_{i,1}$ is taller than or equal to $H_{i+1,2}$. Thus, at least one students of height $H_{i+1,2}$ or $H_{i+1,1}$ will be shorter or equal to the front row's student.
- ✓ As we further expand this idea to $H_{i,3}, H_{i,4}, \cdots, H_{i,M}$, we can prove that $H_{1,j} < H_{2,j} < \cdots < H_{N,j}$ must stand for every j in order to assign every students.
- ✓ If we check the condition after sorting every row by height of students, we can solve the problem of time complexity $\mathcal{O}(NM \log M)$.

1E. Reserved Seat 2

`greedy, sorting, combinatorics`
Difficulty Intention – **Medium**

- Statistics:
  - Senior: 20 Submissions, 8 Accepted, AC rates - 40.000%
- First Solve:
  - Senior: **권찬우**POSTECH, 53 min.
- Writer: 김도훈99asdfg

1E. Reserved Seat 2

- Same as Reserved Seat 1 problem, let $H_{i,j}$ be the height of j-th shortest student in i-th row.
- As seen in the solution of Reserved Seat 1, we can just print 0 if the condition $H_{1,j} < H_{2,j} < \cdots < H_{N,j}$ is not satisfied.
- If the condition above has satisfied, we can confirm that even if only the student in the preceding row is shorter, then every other student in front of them is shorter.
- Therefore, we can calculate every number of cases only by considering the student right in front of them,

1E. Reserved Seat 2

- ✓ Let's assign the students starting from the front row.
- ✓ First row's student can be assigned in any order, so we have $M!$ number of cases to assign.
- ✓ Number of seats where a student with height $H_{2,k}$ can be placed is equal to the number of students with height $H_{i,j} < H_{2,k}$.
- ✓ Since $H_{2,k-1} \leq H_{2,k}$, the seats that can be assigned by the student with height $H_{2,k}$ is always include the seats with height $H_{2,k-1}$.
- ✓ Therefore, we can calculate the number of cases to assign 2-nd row by multiplying $(\text{number of student that satisfy } H_{1,j} < H_{2,k}) - (k-1)$ sequentially.

1E. Reserved Seat 2

- ✓ Number of cases to assign students can be calculated by considering the preceding row only, we can calculate the answer by repeating the multiplication from 2-nd row to N-th row, and multiply by $M!$ which is the number of cases to assign the first row.
- ✓ Number of $H_{i,j} < H_{i+1,k}$ can be found by two pointer or binary search, we can solve the problem by time complexity $\mathcal{O}(NM)$ or $\mathcal{O}(NM \log NM)$.

1C/2D. Double Up

`math, ad_hoc`

Difficulty Intention – **Easy**

- ✓ Statistics:
  - Junior: 39 Submissions, 14 Accepted, AC rates - 35.897%
  - Senior: 42 Submissions, 24 Accepted, AC rates - 57.143%
- ✓ First Solve:
  - Junior: **김재환**POSTECH, 41 min.
  - Senior: **이유담**POSTECH, 10 min.
- ✓ Writer: Shinwook Park`bnb2011`

1C/2D. Double Up

- ✓ Every positive integer can be expressed in the form of $A_i = 2^{a_i} \times b_i$ (b_i is odd).
- ✓ When Dalgoo doubles a number, the value of b_i remains unchanged, and only a_i increases by 1.
- ✓ Therefore, if two numbers A_i and $A_j (A_i < A_j)$ have $b_i = b_j$, then we can make the two numbers equal by applying the operation to A_i $a_j - a_i$ times. Conversely, if $b_i \neq b_j$, we cannot make the two numbers equal no matter how we apply the operation.

- ✓ Now we can see that the **maximum frequency** of the **most frequently occurring number** is equal to the maximum frequency of b_i.
- ✓ Divide all A_i until they are no longer divisible by 2 to obtain b_i. Then, the answer to the problem will be the maximum frequency of b_i.
- ✓ Depending on the implementation, the problem can be solved in $\mathcal{O}\left(N \log \max A_i\right)$ or $\mathcal{O}\left(N \log N \log \max A_i\right)$ time.

2E. UDP Stack

`greedy`, `stack`

Difficulty Intention – **Medium**

- ✓ Statistics:
  - Junior: 61 Submissions, 5 Accepted, AC rates - 8.197%
- ✓ First Solve:
  - Junior: **곽민성**[POSTECH], 59 min.
- ✓ Writer: Leo Jang[`leo020630`]

- ✓ Observation 1.
  - Obviously, having more empty stacks is better.
- ✓ Observation 2.
  - When we open the bottom of a stack with inserted elements, all stored elements to fall down together.
  - So the elements stored in a stack must be contiguous.

- Observation 3.
  - If a closed stack is opened using operation 2, it can be re-opened using operation 2 again.
  - Therefore, it's acceptable to assume that the stack with its bottom opened is always the same. Let's call this stack the U-stack for convenience.
- Based on these facts, we can devise a greedy strategy.

2E. UDP Stack

- Let's assume that we have completed sorting by dropping the numbers up to x through the stack, with x initialized to 0.
- Also, let's denote the front element of permutation A as a, the top element of the D-stack as d, and the top element of the P-stack as p. If the stack is empty, we define the value of d or p as $\emptyset$.

- ✓ 1) $a = x + 1$
  - In this case, inserting a into the U-stack and dropping it immediately is optimal.
- ✓ 2) $a = d + 1$ or $a = p + 1$ $(d, p \neq \emptyset)$
  - In this case, storing a in the corresponding stack is optimal.
- ✓ 3) $d = \emptyset$ or $p = \emptyset$
  - In this case, storing a in an empty stack is optimal.

- If none of the above cases are satisfied and a cannot be inserted, it's impossible to rearrange the permutation in ascending order.
- By repeating this selection process, if at any point the bottom element of either the D-stack or the P-stack is $x + 1$, we simply open the corresponding stack and drop the element.
- And during the process, appropriately change the value of x.
- The time complexity is $\mathcal{O}(N)$ per each test case.

2F. Hyundai Mobis Autonomous Driving Testing 1

`greedy`, `implementation`, `case_work`
Difficulty Intention – **Medium**

- ✓ Statistics:
  - Junior: 37 Submissions, 2 Accepted, AC rates - 5.405%
- ✓ First Solve:
  - Junior: **곽민성**POSTECH, 91 min.
- ✓ Writer: Leo Jang`leo020630`

2F. Hyundai Mobis Autonomous Driving Testing 1

- ✓ Let’s handle the case of impossible to complete driving first.
- ✓ To complete driving is impossible if both cells in a column are obstacles or if there are obstacles arranged diagonally.
- ✓ Note that when $K > 1$, there may be cases where the Nth column and the 1st column of the sample driving track are diagonally connected.

2F. Hyundai Mobis Autonomous Driving Testing 1

- ✓ Let’s calculate the number of operations used for each method.
- ✓ Move straight obviously occurs $NK - 1$ times. Let’s focus on determining the number of lane changes.
- ✓ It’s optimal to drive straight until just before encountering an obstacle and then change lanes.

2F. Hyundai Mobis Autonomous Driving Testing 1

- ✓ Assuming there are T obstacles on the completed demonstration driving track, let's denote the sequence representing the row numbers of the obstacles as $R_1, \cdots, R_T$.
- ✓ The number of occurrences where $R_i \neq R_{i+1}$ is the number of lane changes. Note that due to the large value of K, we cannot directly store the sequence.
- ✓ If we denote the answer for the sequence R representing the sample driving track as X, then $X \times K$ is the final answer. If $R_T \neq R_1$, we need to add $K - 1$ to this.
- ✓ The time complexity is $\mathcal{O}(N)$.

1D. Hyundai Mobis Autonomous Driving Testing 2

`greedy`, `implementation`, `case_work`
Difficulty Intention – **Medium**

- ✓ Statistics:
  - Senior: 93 Submissions, 10 Accepted, AC rates - 10.753%
- ✓ First Solve:
  - Senior: **Mansur Mukimbekov**[UNIST], 41 min.
- ✓ Writer: Leo Jang[`leo020630`]

1D. Hyundai Mobis Autonomous Driving Testing 2

- We basically follow the solution of the Hyundai Mobis Autonomous Driving Testing 1 problem.
- However, since there are multiple types of sample driving tracks, we need to handle more situations.
- During the process of concatenating the sample driving tracks, it's important to note that there may be sample driving tracks without obstacles or even sample driving tracks that are not used at all.
- The time complexity is $\mathcal{O}\left(K + \sum N_i\right)$.

2G. Twitch Plays Pokemon

`dp, graph_traversal`

Difficulty Intention – **Medium**

- ✓ Statistics:
  - Junior: 2 Submissions, 1 Accepted, AC rates - 50.000%
- ✓ First Solve:
  - Junior: **곽민성**[POSTECH], 124 min.
- ✓ Writer: Dongkyu Han[queued_q]

- ✓ Let's say we used the first a commands of Dalgoo and the first b commands of Ponix, and the character ended up at row x and column y.
- ✓ We'll denote this state as (a, b, x, y).
- ✓ The character can then follow either Dalgoo's $a + 1$-th command or Ponix's $b + 1$-th command. For each case, let's call the character's next coordinates (x_D, y_D) and (x_P, y_P), respectively.
- ✓ The next state reachable from (a, b, x, y) is either $(a + 1, b, x_D, y_D)$ or $(a, b + 1, x_P, y_P)$.

2G. Twitch Plays Pokemon

- ✓ We can think of the relation between the states as a graph, and traverse it with BFS.
- ✓ Note that we need to use a 4-dimensional visit- check array over a, b, x, y:
  - Depending on the arrangement orders, there can be various a lot of ways to get to (x, y) using a, b commands, so you need a visit array to avoid reaching the same state again.
  - The escape path may visit the same (x, y) coordinates multiple times. So we need more than just 2-dimensional visit array over.
- ✓ The answer is $a + b$ for the first state that reaches the destination.
- ✓ Let L be the maximum length of the strings. The number of states is N^2L^2, so the total time complexity is $\mathcal{O}\left(N^2L^2\right)$.

2H. Landscape Painting

`dp`, `prefix_sum`
Difficulty Intention – **Hard**

- ✓ Statistics:
  - Junior: 1 Submission, 0 Accepted, AC rates - 0.000%
- ✓ First Solve:
  - Junior: -
- ✓ Writer: Leo Jang[leo020630]

2H. Landscape Painting

- ✓ Let's denote the maximum size of a mountain centered at (i, j) as $M_{i,j}$ and the maximum size of a lake with the top left corner at (i, j) as $L_{i,j}$.
- ✓ There exist mountains of sizes $1, 2, \cdots, M_{i,j}$ centered at (i, j). It is same in the case of lakes.
- ✓ Therefore, by computing $M_{i,j}$ and $L_{i,j}$, we can calculate the answer using a prefix sum technique known as the imos method.

- $M_{i,j}$ and $L_{i,j}$ can be obtained using the following DP.
- $M_{i,j} = \begin{cases} \min(M_{i+1,j-1}, M_{i+1,j}, M_{i+1,j+1}) + 1 & \text{if } S_{i,j} = black \\ 0 & \text{if } S_{i,j} = white \end{cases}$
- $L_{i,j} = \begin{cases} \min(L_{i,j-1}, L_{i,j-1}, L_{i-1,j-1}) + 1 & \text{if } S_{i,j} = white \\ 0 & \text{if } S_{i,j} = black \end{cases}$
- Additionally, there are various other ways to define DP to solve the problem.
- The time complexity is $\mathcal{O}(NM)$.

1F. Moving Walkway

`dijkstra`

Difficulty Intention – **Hard**

- ✓ Statistics:
  - Senior: 19 Submissions, 8 Accepted, AC rates - 42.105%
- ✓ First Solve:
  - Senior: **Mansur Mukimbekov**[UNIST], 67 min.
- ✓ Writer: Leo Jang[`leo020630`]

1F. Moving Walkway

- ✓ Let's consider the moving walkways connected to Building 1.
- ✓ When $u = 1$ (outgoing direction from Building 1):
  - In this case, turning on the moving walkway is always better.
- ✓ When $v = 1$ (incoming direction to Building 1):
  - In this case, turning off the moving walkway is always better since returning to Building 1 is useless.

- ✓ Let's expand this.
- ✓ We use Dijkstra's algorithm. Let's denote the current building being explored as x.
- ✓ Similar to Building 1, turning on the moving walkway in the direction outgoing from Building x and turning off the moving walkway in the direction incoming Building x is always beneficial.
- ✓ Building x is the building with the shortest travel time among the remaining buildings. Therefore, showing that the advancement in the direction that allows returning to Building x is absolutely not beneficial.

1F. Moving Walkway

- Let's denote the shortest travel time to Building i as D_i.
- By assumption, no matter how the moving walkways are set, the travel time to vertex i cannot be shorter than D_i.
- Therefore, this algorithm minimizes the sum of shortest travel times to all buildings.

1F. Moving Walkway

- For each moving walkway, adding directed edges (u, v, d) and $(v, u, 2d)$ to the graph and applying Dijkstra's algorithm will work as described above.
- Determining the power status of each moving walkway is 1 if $D_{u_i} < D_{v_i}$, and 0 otherwise.
- The time complexity is $\mathcal{O}\,(MlogN)$.

1G/2I. Course Registration

`greedy`, `sorting`

Difficulty Intention – **Hard**

- ✓ Statistics:
  - Junior: 0 Submissions, 0 Accepted, AC rates - 0.000%
  - Senior: 19 Submissions, 2 Accepted, AC rates - 10.526%
- ✓ First Solve:
  - Junior: -
  - Senior: **Mansur Mukimbekov**[UNIST], 82 min.
- ✓ Writer: 김도훈[99asdfg]

- ✓ For the courses where $a_i \geq b_i$, it is obvious that Dalgoo will successfully register.
- ✓ After adding such cases to the answer, let's consider only the cases of $a_i < b_i$.
- ✓ Moreover, since we always assume the worst case, we cam assume that other students register according to Dalgoo's priority. In other words, unlike Dalgoo, other students are allowed to change their priorities according to the situation.
- ✓ After some observation, when $a_i < b_i$, we no longer need to consider b_i.
  - This is because in the worst case where Dalgoo registers for the minimum number of courses, there should be no case where other students fail to register.
- ✓ Therefore, further observations can be made independently of b_i.

1G/2I. Course Registration

- ✓ Let the number of remaining courses after excluding the cases be K.
- ✓ Let's find the worst case when Dalgoo's priority is given.
- ✓ From the perspective of other students, it is advantageous not to register for courses that Dalgoo can register for.
- ✓ Therefore, the worst case is when other students fill the capacity of $K - x$ courses, excluding the x courses that Dalgoo can register for according to Dalgoo's priority.
- ✓ In other words, the worst case can always be reduced to finding the minimum x such that the sum of the capacities of the $K - x$ courses that Dalgoo cannot register for is less than or equal to $(M - 1) \times x$.

- From Dalgoo's perspective, we can see that Dalgoo's optimal strategy is to minimize the sum of the capacities of the courses that Dalgoo can register for.
- Let the capacities sorted in ascending order be $a_1, a_2, \cdots, a_K$.
- Dalgoo's optimal strategy is to register courses in ascending order of capacity.
  - If Dalgoo registers for the $(i+1)$-th course before the i-th course, from the perspective of other students, it is the same as reducing the capacity of the courses that Dalgoo cannot register for.
- Therefore, we can see that Dalgoo's optimal strategy is to always register for courses with the smallest capacity first, and other students' optimal strategy is to register for the course with the smallest capacity among the courses that Dalgoo did not register for.

- ✓ Points to note for implementation are as follows:
  - Since Dalgoo is also included in the given M students, note that other students can fill at most $M-1$ capacities per second.
  - In the process of other students filling the capacities of courses with smaller capacities at each second, if they cannot fill the entire capacity at the current time, they should fill the next course first.
  - According to the problem conditions, it is not possible to fail to fill the capacities of two consecutive courses, so you can implement it in a convenient way.
- ✓ By implementing this, the problem can be solved in $\mathcal{O}(N \log N)$ time complexity.

1H. Hyper Tree Problem

`trees`, `disjoint_set`, `bitmask`
Difficulty Intention – **Hard**

- ✓ Statistics:
  - Senior: 7 Submissions, 1 Accepted, AC rates - 14.286%
- ✓ First Solve:
  - Senior: **Mansur Mukimbekov**[UNIST], 133 min.
- ✓ Writer: Shinwook Park[bnb2011]

1H. Hyper Tree Problem

- ✓ Let's consider that the weight range of each edge is $[0, 1]$.
- ✓ When the first type query is given, according to the characteristics of bitwise AND operation, the weight can only change from 1 to 0.
- ✓ Since we restricted the weight range to $[0, 1]$, the possible values for $dist(i, j)$ are either 0 or 1.
- ✓ For $dist(i, j)$ to be 0, all edge weights between vertex i and vertex j must be 0.
- ✓ For $dist(i, j)$ to be 1, there must be at least one edge weight between vertex i and vertex j that is 1.
- ✓ We will focus on the condition for $dist(i, j) = 1$ being easier to handle.

- ✓ When the second type query is given, if we can quickly count the number of vertices i corresponding to $dist(n, i) = 0$, then we can also calculate $\sum_{i=1}^{N} dist(n, i)$.
- ✓ This can be achieved in $\mathcal{O}(1)$ time if we manage the size of the connected components consisting solely of edges with weight 0.
- ✓ The number of edges with weight 0 always increases due to the characteristics of bitwise OR operation. Therefore, by using disjoint set, we can manage the connectivity information of the components in $\mathcal{O}(1)$ time.

1H. Hyper Tree Problem

- ✓ The above approach can be extended to the original weight range as well, leveraging the fact that all operations are bitwise independent.
- ✓ Therefore, by computing the answer for each bit independently and then combining them, queries can be answered in $O(\log W)$ time.
- ✓ The total time complexity is $O(N + Q \log W)$.

11. Ponix and Dalgoo

dp

Difficulty Intention – **Hard**

- Statistics:
  - Senior: 15 Submissions,28 Accepted, AC rates - 13.333%
- First Solve:
  - Senior: **Mansur Mukimbekov**[UNIST], 158 min.
- Writer: Shinwook Park[bnb2011]

- Define $LU[r][c]$ as the maximum weighted sum included in the path from $(1, 1)$ to (r, c).
- Similarly, define $RD[r][c]$ as the maximum weighted sum included in the path from (r, c) to (N, N).
- By using these, we can calculate the maximum weighted sum included in the path from $(1, 1)$ to (N, N) through (r, c).
- This can be computed as $dp[r][c] = LU[r][c] + RD[r][c] - A_{r,c}$.

- Consider the scenario where Dalgoo gives the cells corresponding to rows $[i, i+K-1)$ and columns $[j, j+K-1)$ as a gift to Yunee.
- If Ponix does not pass through this area, then he must necessarily pass through one of the cells $(i+K, 1), (i+K, 2), \cdots, (i+K, j-1)$ or $(i-1, j+K), (i-1, j+K+1), \cdots, (i-1, N)$.
- In this case, the maximum score that Ponix can obtain is the maximum value among $dp[i+K][1], dp[i+K][2], \cdots, dp[i+K][j-1]$ and $dp[i-1][j+K], dp[i-1][j+K+1], \cdots, dp[i-1][N]$.
- By preprocessing the row prefix max and row suffix max of dp, we can compute them in $\mathcal{O}(1)$ time.

- ✓ The number of ways Dalgoo can choose the gift area is $O(N^2)$, and we show that the maximum score of Ponix can obtain in each case can be computed in $\mathcal{O}(1)$ time.
- ✓ Dalgoo wants to minimize Ponix's score, so he selects the area among $O(N^2)$ possibilities where the score Ponix can obtain is minimized.
- ✓ The problem can be solved in $O(N^2)$ time.