```
    2nd LGE Code Jam 2019 - Problem A (Eng

                                                        Pages /... / 2019 Online Code Jam 2 - Editorial
                                                                                                                                                          ☆ Save for later

    2nd LGE Code Jam 2019 - Problem B

                                                        2nd LGE Code Jam 2019 - Problem C(English)

    2nd LGE Code Jam 2019 - Problem B (Eng

                                                        Created by 박선현 sunhyun.park, last modified by 신동철 dc.shin on 2019/07/11

    2nd LGE Code Jam 2019 - Problem C

                                                        Stock Market

    2nd LGE Code Jam 2019 - Problem C(En

    2nd LGE Code Jam 2019 - Problem D

                                                        This is a problem to find the best suitable minimum length of range which has sum greater or equal than some number.

    2nd LGE Code Jam 2019 - Problem D (Eng

                                                        The search space N is maximum 10<sup>5</sup> size, and time limit is 1 second.
          • 2nd LGE Code Jam 2019 - Problem E

    2nd LGE Code Jam 2019 - Problem E (Eng

                                                        Brute Force - O(N^2)

    2nd LGE Code Jam 2019 - Problem F

    2nd LGE Code Jam 2019 - Problem F (Eng

                                                        This approach is searching every range which has sum greater or equal than given number.
> 코드잼 준비하기
 • 2019년 활동
                                                        We can use prefix sum data structure to compute sum of continuous range.
2018년 활동
                                                        By using prefix sum data structure, we can get the sum of continuous range in array at O(1) time complexity.
> 2016년 활동
                                                            sum of range [i,j] = prefix_sum [j] - prefix_sum [i-1]
2015년 활동
> 2014년 활동
                                                        Actual benefit can get for each day can be obtained by subtracting purchasing price from selling price.
2013년 활동
                                                        We're using this value to compute actual benefit and make prefix sum.
> 2012년 활동
                                                        The search space limitation for subtask1 is N<=5000. And this is small enough for naive O(N^2) approach to solve it within given time bound.
▶ 코딩전문가와 함께하는 코딩 도장
 • 코딩 전문가가 참여하는 커뮤니티 모임
                                                          C-sub1-solution.cpp
♣ Space tools ▼
                                                                  #include <algorithm>
                                                                  #include <iostream>
                                                                  #include <vector>
                                                                  using ll = long long;
                                                                  using pii = std::pair<int, int>;
                                                                  using namespace std;
                                                                  void solve() {
                                                                      int n, current_cash, cost;
                                                            11
                                                                      11 profit_goal;
                                                            12
                                                            13
                                                                      cin >> n >> current_cash >> cost >> profit_goal;
                                                            14
                                                                      vector<ll> profit(n+1, 0);
                                                            15
                                                                      for (int i = 1; i <= n; ++i) {</pre>
                                                            16
                                                            17
                                                                          cin >> profit[i];
                                                                          profit[i] += profit[i - 1] - cost;
                                                            18
                                                            19
                                                            20
                                                                      int can_have = 0;
                                                            21
                                                                      if (cost) {
                                                            22
                                                            23
                                                                          can_have = current_cash / cost;
                                                                      } else {
                                                            24
                                                            25
                                                                          can_have = n;
                                                            26
                                                            27
                                                            28
                                                                      pii best = {n + 1, 0};
                                                                      for (int i=0; i<n; ++i) {</pre>
                                                            29
                                                            30
                                                                          for (int j=i+1; j<=n; ++j) {</pre>
                                                                              if (profit[j] - profit[i] >= profit_goal) {
                                                            31
                                                                                   best = min(best, pii(j - i, -i));
                                                            32
                                                            33
                                                            34
                                                            35
                                                            36
                                                            37
                                                                      if (best.first > can_have) {
                                                                          cout << -1 << endl;</pre>
                                                            38
                                                            39
                                                                      } else {
                                                                          cout << -best.second + 1 << " " << -best.second + best.first << endl;</pre>
                                                            40
                                                            41
                                                            42
                                                            43
                                                                 int main(int argc, char* argv[]) {
                                                                      ios::sync_with_stdio(false);
                                                            45
                                                                      cin.tie(0);
                                                            46
                                                                      cout.tie(0);
                                                            47
                                                                      int tcs;
                                                            48
                                                                      cin >> tcs;
                                                            49
                                                            50
                                                                      while (tcs-- > 0) solve();
                                                            51
                                                            52
                                                          C-sub1-solution.java
                                                                  import java.util.*;
                                                                  public class Main {
                                                                      private static int T;
                                                                      private static int N;
                                                                      private static long X; // money
                                                                      private static long Y; // price
                                                                      private static long Z; // min
                                                                      public static void main(String args[]) {
                                                            10
                                                                          Scanner sc = new Scanner(System.in);
                                                            11
                                                                          T = sc.nextInt();
                                                            12
                                                                          for (int i = 0; i < T; i++) {</pre>
                                                            13
                                                                              N = sc.nextInt();
                                                            14
                                                                              X = sc.nextLong();
                                                            15
                                                                              Y = sc.nextLong();
                                                            16
                                                                              Z = sc.nextLong();
                                                            17
                                                                              long[] Vs = new long[N+1];
                                                            18
                                                                              for (int j = 1; j <= N; j++) {</pre>
                                                            19
                                                                                   Vs[j] = Vs[j - 1] + sc.nextLong() - Y;
                                                            20
                                                            21
                                                                               solve(Vs);
                                                            22
                                                            23
                                                                          sc.close();
                                                            24
                                                            25
                                                            26
                                                                      private static void solve(long[] Vs) {
                                                            27
                                                                          long start = 0;
                                                            28
                                                                          long range = N + 1;
                                                            29
                                                                          for (int i = 0; i <= N; i++) {</pre>
                                                            30
                                                                              for (int j = i + 1; j <= N; j++) {</pre>
                                                            31
                                                                                   if (Vs[j] >= Vs[i] + Z) {
                                                            32
                                                                                       if (range > j - i || (range == j - i && i >= start)) {
                                                            33
                                                                                           start = i + 1;
                                                            34
                                                                                           range = j - i;
                                                            35
                                                            36
                                                            37
                                                            38
                                                            39
                                                                          if (range == N + 1) {
                                                            40
                                                                               System.out.println("-1");
                                                            41
                                                                          } else {
                                                            42
                                                                               System.out.println(start + " " + (start + range - 1));
                                                            43
                                                            44
                                                            45
                                                            46
                                                        <u>Using sliding window and 2-pointers - O(N)</u>
                                                        Search space limitation for subtask2 is N<=10^5. O(N^2) algorithm may have time limit exceeding failure.
                                                        At least, O(N log N) or faster algorithm can be a solution.
                                                        Is there a way to find the possible range with this time complexity? There is a method which called "Sliding Window".
                                                        "Sliding Window" is a method to compensate time complexity only checks the differences while sliding interesting range.
                                                        In general, sliding window use fixed size window to compute range operations,
                                                        However in this problem, we're going to use variable length sliding window with two pointers(these pointers refer starting & ending accordingly) technique.
                                                        Below is an example of finding range which has sum at least 10 using slinding window with two pointers.
                                                           <Problem> Finding subarray sum at least 10 from a positive array
                                                            Initial step
                                                            (move right point first and then move left point until meet the boundary),
                                                                   RIGHT
                                                                                                                                     RIGHT
                                                                                                                                                                                      RIGHT
                                                                          3 2 2 5 1
                                                                                                                     LEFT
                                                                                                                                                                         LEFT
                                                                    LEFT
                                                            Same steps for further processing,
                                                                                        RIGHT
                                                                                                                                                 RIGHT
                                                                                                                                                                                                     RIGHT
                                                                                                                               LEFT
                                                                          LEFT
                                                                                                                                                                                    LEFT
                                                        In this method, starting & ending points are advancing only one direction. So, total amount of scanning is 2 times of size of array.
                                                        As we can get range sum in O(1) with prefix sum, this method has O(N) time complexity.
                                                        However, we have negative values in our array.
                                                        As this method only valid for positive array, we can not using this method directly to our problem.
                                                        So, should we consider another approach to solve this problem?
                                                        Let's think about a little. We can found the fact that the range starting or ending with negative value can not be optimal.
                                                        This is because we can always have better(has small length and has bigger sum) range without these negative values.
                                                        Let's checks some more details in this point of view.
                                                        (It is very important to check conditions for an algorithm before use it)
                                                         About starting point x, Ending point y and prefix sum array P
                                                          - optimal(y) : x for x<=y and the smallest length [x,y] with P[y]-P[x-1]>=Z (P[x-1]<=P[y]-Z)
                                                          - If x1<x2 and P[x2-1] <= P[x1-1], optimal(y) can not be x1.
                                                           Because P[y] - P[x1-1] \ge Z, shortly P[x1-1] \le P[y] - Z then,
                                                           We can see P[x2-1] \le P[x1-1] \le P[y] - Z
                                                           And in this case, x1 is a better choice comparing with x2 because it is smaller in size.
                                                           So, if ending point is determined with y, we can only consider x for increasing P[x].
                                                          - At the same point of view, the ending points y1 < y2,
                                                          If optimal(y1)==x, then we don't need to reconsider x at checking y2(It is always [x,y1] is smaller in size)
                                                        Generally speaking, when using sliding window with two pointers we should maintain monotonic queue while sliding.
                                                        In this problem, adding index to queue should always strictly increasing total amount of queue.
                                                        And removing index from window should always strictly decreasing total amount of queue as well.
                                                        If there're some indexes which are not appropriate for this, then we don't need to consider them.
                                                        We just remove them greedily, and maintaining monotonic queue while processing.
                                                        (As we are computing range sum with prefix sum array, removing some indexes between starting & ending doesn't have effect for this)
                                                        The workflow is shown below.
                                                         <Problem> Finding minimum length of subarray sum at least 10 in an array
                                                              we're working on prefixsum array,
                                                                                                              prefixsum
                                                                                                                                                          15
                                                                                                                                                               16 22
                                                                                          -3
                                                                                                   6
                                                                                                                                                 10 18
                                                              Initial step
                                                              (move right point first and then move left point until meet the boundary)
                                                                                                                                             RIGHT
                                                                RIGHT
                                                                                       18
                                                                                           15
                                                                                                    22
                                                                                                                                                      15
                                                                                                                                                           16
                                                                                                                                                               22
                                                                                               16
                                                                                                         25
                                                                LEFT
                                                                                                                                                                          Note that as we're using
                                                                                                                           LEFT
                                                                                                                                                                          monotonic queue,
                                                                                                                                                                          some indexes are not
                                                                                                                                                                          considered for answer
                                                               Same steps for further processing,
                                                                                                                                                                          (referencing minus value in
                                                                                                                                                                          original array)
                                                                                                                                                                  RIGHT
                                                                                    RIGHT
                                                                                                                                                               22
                                                                                           15 16 22
                                                                                      18
                                                                                                         25
                                                                                                                                                   LEFT
                                                                              LEFT
                                                        When implementing the algorithm, we should insert and delete to both side of queue.
                                                        So, it is better to use deque or list style data structure for faster computation.
                                                          C-sub2-solution.cpp
                                                                  #include <algorithm>
                                                                  #include <deque>
                                                                  #include <iostream>
                                                                  #include <vector>
                                                                  using ll = long long;
                                                                  using pii = std::pair<int, int>;
                                                                  using namespace std;
                                                            10
                                                                  void solve() {
                                                                      11 n, current_cash, cost, profit_goal;
                                                            12
                                                                      cin >> n >> current_cash >> cost >> profit_goal;
                                                            13
                                                            14
                                                            15
                                                                      11 can_have = cost ? current_cash / cost : n;
                                                            16
                                                                      vector<ll> profit(n + 1, 0);
                                                            17
                                                                      for (int i = 1; i <= n; ++i) {</pre>
                                                            18
                                                                          cin >> profit[i];
                                                            19
                                                            20
                                                                          profit[i] -= cost;
                                                            21
                                                            22
                                                                          // making cumulative
                                                                          profit[i] += profit[i - 1];
                                                            23
                                                            24
                                                            25
                                                            26
                                                                      pii best = {n + 1, 0};
                                                                      deque<11> dq;
                                                            27
                                                                      for (int i = 0; i <= n; ++i) {</pre>
                                                            28
                                                                          while (!dq.empty() && profit[i] <= profit[dq.back()]) dq.pop_back();</pre>
                                                            29
                                                            30
                                                                          while (!dq.empty() && profit[i] >= profit[dq.front()] + profit_goal) {
                                                            31
                                                                              int 1 = dq.front();
                                                            32
                                                            33
                                                                              dq.pop_front();
                                                                               best = min(best, pii(i-1, -1));
                                                            34
                                                            35
                                                             36
                                                            37
                                                                          dq.push_back(i);
                                                            38
                                                            39
                                                                      if (best.first > can_have) {
                                                            40
                                                                          cout << -1 << endl;</pre>
                                                            41
                                                                      } else {
                                                            42
                                                                          cout << -best.second + 1 << " " << -best.second + best.first << endl;</pre>
                                                            43
                                                            44
                                                            45
                                                            46
                                                                 int main(int argc, char* argv[]) {
                                                                      ios::sync_with_stdio(false);
                                                            48
                                                                      cin.tie(0);
                                                            49
                                                                      cout.tie(0);
                                                            50
                                                            51
                                                                      int tcs;
                                                                      cin >> tcs;
                                                            52
                                                            53
                                                            54
                                                                      while (tcs-- > 0) solve();
                                                            55
                                                          C-sub2-solution.java
                                                                  import java.util.*;
                                                                 public class Main {
                                                                      private static int T;
                                                                      private static int N;
                                                                      private static long X; // money
                                                                      private static long Y; // price
                                                                      private static long Z; // min
                                                                      public static void main(String args[]) {
                                                            10
                                                                          Scanner sc = new Scanner(System.in);
                                                            11
                                                                          T = sc.nextInt();
                                                                          for (int i = 0; i < T; i++) {</pre>
                                                                               N = sc.nextInt();
                                                            14
                                                                              X = sc.nextLong();
                                                            15
                                                                              Y = sc.nextLong();
                                                            16
                                                                              Z = sc.nextLong();
                                                            17
                                                                              long[] Vs = new long[N+1];
                                                            18
                                                                              for (int j = 1; j <= N; j++) {</pre>
                                                            19
                                                                                   Vs[j] = Vs[j - 1] + sc.nextLong() - Y;
                                                            20
                                                            21
                                                                               solve(Vs);
                                                            22
                                                            23
                                                                          sc.close();
                                                            24
                                                            25
                                                            26
                                                                      private static void solve(long[] Vs) {
                                                            27
                                                                          long start = 0;
                                                            28
                                                                          long range = N + 1;
                                                            29
                                                                          Deque<Integer> deq = new LinkedList<>();
                                                            30
                                                                          for (int i = 0; i <= N; i++) {</pre>
                                                            31
                                                                               while (!deq.isEmpty() && Vs[i] <= Vs[deq.getLast()]) {</pre>
                                                            32
                                                                                   deq.removeLast();
                                                            33
                                                            34
                                                                              while (!deq.isEmpty() && Vs[i] >= Vs[deq.getFirst()] + Z) {
                                                            35
                                                                                   int s = deq.removeFirst();
                                                            36
                                                                                   if (range > i - s || (range == i - s && s >= start)) {
                                                            37
                                                                                       start = s + 1;
                                                            38
                                                                                       range = i - s;
                                                            39
                                                            40
                                                            41
                                                                              deq.addLast(i);
                                                            42
                                                            43
                                                                          if (range == N + 1) {
                                                            44
                                                                               System.out.println("-1");
                                                            45
                                                                          } else {
                                                            46
                                                                               System.out.println(start + " " + (start + range - 1));
                                                            47
                                                            48
                                                            50
                                                        🖒 Like 배성호 baver.bae likes this
                                                                                                                                                                                         No labels 🥒
                                                               Write a comment...
```

1 LG Electronics

ద్ద COLLAB

Spaces ▼

People Questions

Calendars

Create •••