1. D - Enough Array [AtCoder] : Two Pointer Problem


ans = (# of subarr (sum < k))

$$\left[\frac{x(x+1)}{2} - \text{Ans}\right]$$

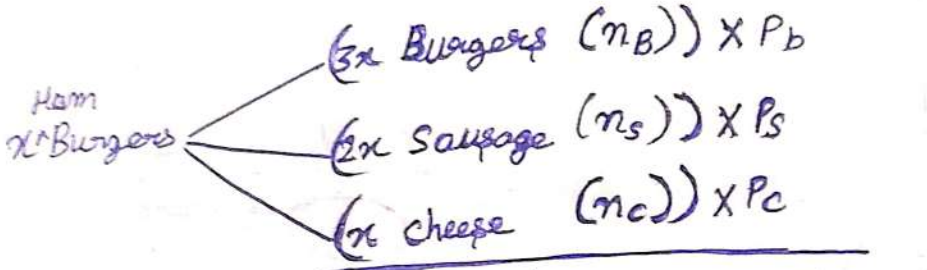
Total − smaller than k through variable window

Pseudocode :-

```
sum = 0, head = -1, tail = 0
while (tail < n)
    While ((head + 1 < n) && (arr(head+1) + sum) < k)
        head ++;
        sum += arr[head];
    ans -= head - tail + 1
    if (tail < head) {
        sum -= arr[tail]
        tail ++;
    }
else {
    tail ++;
    head = tail - 1
}
```

2. Hamburgers [Codeforces 371C]

Min/Max → BS / Greedy / DP

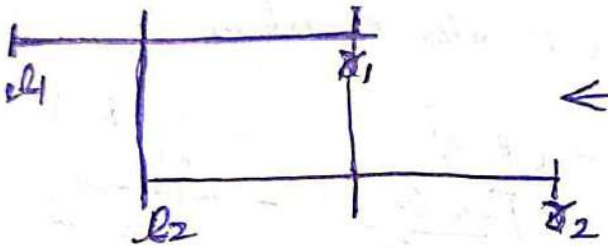


```cpp
#include<bits/stdc++.h>
using namespace std;

#define endl '\n'
using lli = long long int;
lli nb_have, ns_have, nc_have;
lli pb_price, ps_price, pc_price;
lli budget;
lli req_b = 0, req_s = 0, req_c = 0;
bool check(lli num_hamburgers) {
    lli total_b_needed = num_hamburgers * req_b;
    lli total_s_needed = num_hamburgers * req_s;
    lli total_c_needed = num_hamburgers * req_c;

    lli buy_b = max(0LL, total_b_needed - nb_have);
    lli buy_s = max(0LL, total_s_needed - ns_have);
    lli buy_c = max(0LL, total_c_needed - nc_have);
    lli total_cost = buy_b * pb_price + buy_s * ps_price + buy_c * pc_price;
    return total_cost <= budget;
}

void solve() {
    string recipe;
    cin >> recipe;
    cin >> nb_have >> ns_have >> nc_have;
    cin >> pb_price >> ps_price >> pc_price;
    cin >> budget;

    for (char c : recipe) {
        if (c == 'B') req_b++;
        else if (c == 'S') req_s++;
        else req_c++;
    }
    lli low = 0;
    lli high = 2e14;
    lli ans = 0;

    while (low <= high) {
        lli mid = low + (high - low) / 2;
        if (check(mid)) {
            ans = mid;
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }
    cout << ans << endl;
}
```

3. The meeting place cannot be changed [Codeforces – 780 B]

Binary Search
↳ Binary Search on (Ans) → time


$i^{th}$ friend can be anywhere in $[x_i - v_i * t, x_i + v_i * t]$.




Intersection
$\leftarrow [\max(l_1, l_2), \min(r_1, r_2)]$

All friends can only meet if the intersection of intervals is non-empty.
If time $t$ is enough so any time $(>t)$ is also enough. so the predicate "can_meet(t)" is monotonic ⇒ we can do binary search on time.

```cpp
bool check(double time) {
    double max_left_bound = 0.0;
    double min_right_bound = 1e10;

    for (int i = 0; i < n; i++) {
        max_left_bound = max(max_left_bound, (double)x[i] - (double)v[i] * time);
        min_right_bound = min(min_right_bound, (double)x[i] + (double)v[i] * time);
    }

    return max_left_bound <= min_right_bound;
}

void solve() {
    cin >> n;
    x.resize(n);
    v.resize(n);

    for (int i = 0; i < n; i++) cin >> x[i];
    for (int i = 0; i < n; i++) cin >> v[i];

    double low = 0.0;
    double high = 1e9 + 7;

    for(int i = 0; i < 100; i++) {
        double mid = low + (high - low) / 2.0;
        if (check(mid)) {
            high = mid;
        } else {
            low = mid;
        }
    }

    cout << fixed << setprecision(12) << high << endl;
}
```

4. Quiz Master → Two pointers or Binary Search

1. Sort the students by smartness → [a1, a2 ---- an]
2. min smartness = a[L]
   max " = a[R]

Goal = minimize the gap (a[R] - a[L])

Sort smartness, then use sliding window to find smallest interval [L, R] where student covers all topics.