T3 - Communication 题解

题目简述

题目大意：
给定三个长度为 N 的序列 a, b, w 和两个参数 L, R。
构造一张有向图，点 $u \to v$ 有边当且仅当 $L \le a_u + b_v \le R$。
求从起点出发，到所有点的最短路。

- $N \le 2 \times 10^5$。
- 点权 $w_i \ge 0$。
- 最短路定义为路径上所有点的点权之和。

The Key：
这是一个**隐式建图的最短路问题**。由于边数可能达到 $O(N^2)$，我们不能直接建图，而必须利用边存在的条件 $b_v \in [L - a_u, R - a_u]$ 来优化。

子任务分析

Subtask 1：$N \le 7$

由于 N 极小，可以直接搜索所有可能的路径或枚举全排列，判断合法性。

时间复杂度：$\mathcal{O}(N!)$ 或 $\mathcal{O}(2^N \cdot N)$。

Subtask 2：$a_1 = a_2 = \cdots = a_N$

当所有 a_i 相等时，每个点发出的边所要求的 b_v 范围是一模一样的。
这意味着如果点 0 能到 v，那么任何能被 0 到的点 u 也能到 v。
最短路非常简单：满足 $L \le a_0 + b_v \le R$ 的点 v 距离为 $w_0 + w_v$，其余不可达。

时间复杂度：$\mathcal{O}(N)$

Subtask 3: $N \leq 10^3$

边数最多 10^6，可以显式枚举所有对 (u, v)，若满足条件则建边。
最后跑一次 `Dijkstra` 算法。

时间复杂度：$\mathcal{O}(N^2 \log N)$ 或 $\mathcal{O}(N^2)$。

Subtask 4: $L = 1$

条件简化为 $1 \leq a_u + b_v \leq R$。在 a, b 为正数的情况下，左侧通常恒成立，只需考虑 $b_v \leq R - a_u$。
这是一个前缀连边问题。我们可以将点按 b_i 升序排序，每个点连向比它 b 值小的下一个点（形成一条链），这样只需要连 $O(N)$ 条边即可表达所有前缀关系。

时间复杂度：$\mathcal{O}(N \log N)$

Subtask 5: $N \leq 5 \times 10^4$

可以使用**线段树优化建图**。将点按 b_i 排序后，每个点 u 连向的是 b 值在某个区间内的点。线段树可以将区间连边优化到 $O(\log N)$。

时间复杂度：$\mathcal{O}(N \log N)$

正解

Dijkstra + 平衡树/Set 优化

我们注意到 `Dijkstra` 的性质如下：

> 在 Dijkstra 算法执行过程中，由于点权 $w_i \geq 0$，一旦一个点被从优先队列中取出，它的最短路就已经确定了。

对于当前确定的最短路点 u，我们需要找到所有**尚未访问**过的点 v，满足：
$L - a_u \leq b_v \leq R - a_u$

由于每个点只需要被访问（松弛）一次，我们不需要预先建边。我们可以把所有还没找到最短路点的点放进一个按 b_i 排序的 `std::set` 中。

起点距离为 w_0，加入优先队列。其余点全部放入 `set`。从优先队列取出当前距离最小的点 u。在 `set` 中利用 `lower_bound` 找到第一个 $b_v \geq L - a_u$ 的点。

从该位置开始向后遍历，只要 $b_v \le R - a_u$，就更新 v 的距离，将 v 加入优先队列，并**从 `set` 中永久删除** v。每个点最多进出 `set` 一次，进出优先队列一次。

时间复杂度：$\mathcal{O}(N \log N)$。

```cpp
//官方 std
#include <bits/stdc++.h>
#define uwu return 0;

using namespace std;

#define fs first
#define sc second

int main(){
    cin.tie(0), ios::sync_with_stdio(0);
    int N, L, R;
    cin >> N >> L >> R;

    vector<int> a(N), b(N);
    vector<long long> w(N);

    for(auto &i:a){
        cin >> i;
    }

    for(auto &i:b){
        cin >> i;
    }

    for(auto &i:w){
        cin >> i;
    }

    set <pair<int, int>> sort_by_b;
    priority_queue <pair<long long, int>> pq;

    vector <long long> dist(N);

    for (int i = 1; i < N; i++){
        sort_by_b.insert({b[i], i});
    }
    for(auto &i:dist){
```

```
        i = -1;
    }
    pq.push({-w[0], 0});

    while(!pq.empty()){
        pair <long long, int> tp = pq.top();
        pq.pop();
        dist[tp.sc] = -tp.fs;
        for (auto it = sort_by_b.lower_bound({L - a[tp.sc], 0});
it ≠ sort_by_b.end() && it→fs + a[tp.sc] ≤ R; it =
sort_by_b.erase(it)){
            pq.push({tp.fs - w[it→sc], it→sc});
        }
    }
    for(auto i:dist){
        cout << i << ' ';
    }
    cout << '\n';
    uwu;
}
```