Contest Duration: 2025-05-24(Sat) 08:00 (http://www.timeanddate.com/worldclock/fixedtime.html?iso=20250524T2100&p1=248) - 2025-05-24(Sat) 09:40 (http://www.timeanddate.com/worldclock/fixedtime.html?iso=20250524T2240&p1=248) (local time) (100 minutes)

Back to Home (/home)

Top (/contests/abc407) Tasks (/contests/abc407/tasks)

Clarifications (/contests/abc407/clarifications) Submit (/contests/abc407/submit)

Results Standings (/contests/abc407/standings)

Virtual Standings (/contests/abc407/standings/virtual)

Custom Test (/contests/abc407/custom_test) Editorial (/contests/abc407/editorial)

Discuss (https://codeforces.com/blog/entry/143155)

Official

E - Most Valuable Parentheses (/contests/abc407/tasks/abc407_e) Editorial by **en_translator** (/users/en_translator)

Among various equivalent definitions of a correct parenthesis sequence, one of the most intuitive one is:

- For $1 \leq i \leq 2N$, among $S_1, \dots \dots . S_i$, at least $\frac{i}{2}$ characters are `(`.
- Among $S_1, \dots, S_{2N}$, exactly N characters are `(`.

Taking the "$\frac{i}{2}$" as the variable, we can further rephrase it like this:

- For all $1 \leq k \leq N$, among $S_1, \dots, S_{2k-1}$, at least k characters are `(`.
- Among $S_1, \dots, S_{2N}$, exactly N characters are `(`.

This definition suggests the following procedure to construct a parenthesis sequence:

1. Determine $S_1 =$ `(`. (Regard this step as $k = 1$ for convenience.)
2. For $2 \leq k \leq N$ in ascending order:
    - Add $2k-2, 2k-1$ to the set of candidates.
    - Choose one element x from the set of candidates and remove it. Determine $S_x =$ `(`.
3. Set the undetermined N characters to be `(`.

This procedure indeed yields a correct parenthesis sequence, and any correct parenthesis sequence can be constructed by this procedure.

2025-05-24 (Sat)
23:46:52 -04:00

Proof that the parenthesis sequences generated by this procedure are equal to the correct parenthesis sequences

Take a sequence S constructed by this procedure. For all $1 \leq t \leq N$,

- For all k with $1 \leq k \leq t$, at least one of $S_1, \ldots, S_{2t-1}$ is set `(`, so at least t characters among $S_1, \ldots, S_{2t-1}$ is `(`.
- There are exactly N `(`'s.

Hence, S is a correct parenthesis sequence.

Conversely, take a correct parenthesis sequence S. For all $1 \leq t \leq N$,

- S_1 is `(`.
- At least t characters among $S_1, \ldots, S_{2t-1}$ is `(`. When $k = t$, only $(t-1)$ elements have been chosen to be `(`, so there is at least one element that can be chosen for $k = t$.
- There are exactly N `(`'s.

Hence, S can be constructed by the procedure. **(End of proof)**

Here, one can prove as follows that the final score of S can be maximized by, when choosing an element x from the set of candidate, picking the one that maximizes A_x.

The proof that the greedy algorithm is valid

Suppose that when $k = t$, while A_x is the maximum element, we instead choose $A_y (< A_x)$.

- If A_x is not chosen for any $k > t$: by picking A_x instead of A_y and choosing the same elements for the other k, the procedure is still valid, but the score of S increases.
- If A_x is chosen for some $k = t' > t$: by choosing A_x for $k = t$ and choosing A_y for $k = t'$, while choosing the same elements for the other k, the procedure is still valid, but the score of S increases.

In any case, there is a better choice to improve the final score of S. Therefore, choosing $A_y \ (< A_x)$ for $k = t$ is not optimal, and thus choosing A_x is optimal. **(End of proof)**

The implementation can be simplified by using a priority queue. (C++ has a standard library `std::priority_queue`.)

The sample code in C++ is shown below.

Copy

2025-05-24 (Sat)
23:46:52 -04:00

```cpp
#include <iostream>
#include <vector>
#include <queue>
using std::cin;
using std::cout;
using std::cerr;
using std::endl;
using std::vector;
using std::priority_queue;
using std::greater;


typedef long long ll;

ll solve (const ll n, const vector<ll> &a) {
        ll ans = 0;

        priority_queue<ll, vector<ll> > que;
        for (ll i = 0; i < n; i++) {
                if (i == 0) {
                        que.push(a[i*2-0]);
                } else {
                        que.push(a[i*2-1]);
                        que.push(a[i*2-0]);
                }

                ll v = que.top();
                que.pop();

                ans += v;
        }

        return ans;
}

int main (void) {
        int T;
        cin >> T;
        while (T--) {
                ll n;
                cin >> n;
                vector<ll> a(n*2);
                for (ll i = 0; i < n*2; i++) {
                        cin >> a[i];
                }

                cout << solve(n, a) << "\n";
```

```
        }

        return 0;
}
```

posted: about an hour ago

last update: about an hour ago

am)

ps%3A%2F%2Fatcoder.jp%2Fcontests%2Fabc407%2Feditorial%2F13107%3Flang%3Den&title=Editorial%20-
)407)

2025-05-24 (Sat)
23:46:52 -04:00