Zadanie A
Asymilacja

Zgłoszeń: 84
Zaakceptowanych: 36

Pierwsze rozwiązanie:
uj8 [Kapała, Tokarz, Deka]
00:21:23

Autor: Lech Duraj

Mamy n planet do opanowania, o wartościach $a_1, \ldots, a_n$ oraz k statków asymilacyjnych. W jednym ruchu możemy dokonać inwazji na planetę o wartości m, lądując na niej $s \geq m$ statkami. Planeta robi się wtedy opanowana, a jej nowa wartość to $m + s$. Z takiej planety można w przyszłości (ale tylko raz) podnieść $m + s$ statków. Ile co najmniej razy trzeba podnieść, by podbić wszystko?

Mamy n planet do opanowania, o wartościach $a_1, \ldots, a_n$ oraz k statków asymilacyjnych. W jednym ruchu możemy dokonać inwazji na planetę o wartości m, lądując na niej $s \geq m$ statkami. Planeta robi się wtedy opanowana, a jej nowa wartość to $m + s$. Z takiej planety można w przyszłości (ale tylko raz) podnieść $m + s$ statków. Ile co najmniej razy trzeba podnieść, by podbić wszystko?

Zadanie inspirowane mechaniką migracji barbarzyńców w grze *Imperator: Rome*.

Mamy n planet do opanowania, o wartościach $a_1, \ldots, a_n$ oraz k statków asymilacyjnych. W jednym ruchu możemy dokonać inwazji na planetę o wartości m, lądując na niej $s \geq m$ statkami. Planeta robi się wtedy opanowana, a jej nowa wartość to $m + s$. Z takiej planety można w przyszłości (ale tylko raz) podnieść $m + s$ statków. Ile co najmniej razy trzeba podnieść, by podbić wszystko?

Zadanie inspirowane mechaniką migracji barbarzyńców w grze *Imperator: Rome*.

(dobrze zastosowana poniższa strategia = lawina barbarzyńców!)

Jeśli liczba statków na początku starczy na opanowanie wszystkich planet od razu, to odpowiedź jest oczywiście 0.

Jeśli liczba statków na początku starczy na opanowanie wszystkich planet od razu, to odpowiedź jest oczywiście 0.
Ale co, jeśli nie jest?

Jeśli liczba statków na początku starczy na opanowanie wszystkich planet od razu, to odpowiedź jest oczywiście 0.
Ale co, jeśli nie jest?

Na pewno jest jakaś planeta, którą będziemy mobilizować. Co więcej, na pewno jest planeta P, którą będziemy mobilizować jako pierwszą.

Jeśli liczba statków na początku starczy na opanowanie wszystkich planet od razu, to odpowiedź jest oczywiście 0.
Ale co, jeśli nie jest?

Na pewno jest jakaś planeta, którą będziemy mobilizować. Co więcej, na pewno jest planeta P, którą będziemy mobilizować jako pierwszą.

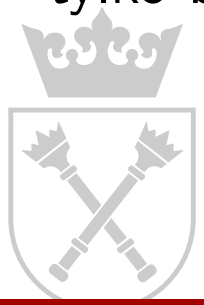
Ale w takim razie można *od razu* opanować P i *od razu* ją zmobilizować – dalej będzie tylko łatwiej.

Jeśli liczba statków na początku starczy na opanowanie wszystkich planet od razu, to odpowiedź jest oczywiście 0.
Ale co, jeśli nie jest?

Na pewno jest jakaś planeta, którą będziemy mobilizować. Co więcej, na pewno jest planeta P, którą będziemy mobilizować jako pierwszą.

Ale w takim razie można *od razu* opanować P i *od razu* ją zmobilizować – dalej będzie tylko łatwiej.

Czyli wiemy, że pierwszym naszym ruchem będzie opanowanie jakiejś planety i natychmiastowa mobilizacja. To teraz oczywiste jest, którą planetę wybrać – największą, na jaką możemy się porwać. Inny wybór tylko by nam utrudnił sprawę, bo mielibyśmy dalej mniej statków.



Pełny algorytm wygląda tak:

Q – zbiór jeszcze nie opanowanych planet
s – suma wartości planet w Q
k – liczba naszych statków
wynik – liczba mobilizacji, jakie wykonaliśmy.

- Jeżeli $k \geq s$, to opanuj wszystkie planety i zakończ algorytm.
- W przeciwnym razie, weź p – największą planetę z Q mniejszą lub równą k.
- $Q = Q - \{p\}$; $s = s - p$; $k = k + p$; $wynik = wynik + 1$.

Pełny algorytm wygląda tak:

Q – zbiór jeszcze nie opanowanych planet
s – suma wartości planet w Q
k – liczba naszych statków
wynik – liczba mobilizacji, jakie wykonaliśmy.

- Jeżeli $k \geq s$, to opanuj wszystkie planety i zakończ algorytm.
- W przeciwnym razie, weź p – największą planetę z Q mniejszą lub równą k.
- $Q = Q - \{p\}$; $s = s - p$; $k = k + p$; $wynik = wynik + 1$.

Najprościej implementować Q jako `multiset`, co daje algorytm $\mathcal{O}(n \log n)$
Ale można też użyć posortowanej tablicy – nieco dłuższy kod, nieco lepsza stała w złożoności.

```
multiset<int> s{10, 20, 30};
long long x = 123000123000LL;
auto it = s.lower_bound(x);
if(it != s.end()) {
    cout << *it;
}
```

(ten kod wypisze liczbę 10)