Анализ на задача `belote`

Тагове: оптимизация с bitset, техниката small-to-large

Решение на първа подзадача – 20 точки

Достатъчно е директно да симулираме игрите по дни, както е описано в условието. Можем да използваме *set* или булев масив, за да пазим за всеки приятел с кои други трябва да проведе игри. За улеснение като добавяме допълнителна игра, можем да я добавяме само към приятеля с по-малкия номер, защото всяка игра се играе в деня на приятеля с по-малък номер. Отговорът е сумата от големините на множествата от всеки ден.

Сложност: $O(N^3 \log N)$ със *set* или $O(N^3)$ с булев масив.

Решение на втора подзадача – 50 точки (=20+30)

Проблем на горното решение е, че макар и на теория максималният брой игри, които могат да се играят, да са $\frac{N(N-1)}{2}$, някои игри ще ги добавяме по много пъти. Например, ако първия ден приятел 1 има игри с $2, 3, 4$, това предизвиква игри $(2, 3), (2, 4), (3, 4)$, но игра $(3, 4)$ ще я разгледаме като наложена и във втория ден, защото приятел 2 ще има игри с 3 и 4. Затова вместо за всеки приятел да поддържаме множеството от други приятели, с които трябва да играе, ще поддържаме множества, в които всеки трябва да играе с всеки от следващите (ако приемем, че множеството е наредено в нарастване на номерата). Съответно всяко такова множество ще го запазваме за приятеля с най-малкия номер. Лесно е да се види, че ако трябва да добавим второ множество за даден приятел, то можем да обединим двете множества, защото когато се играят неговите игри, то ще получим игри между всички от останалите в двете множества.

Така поддържаме множества S_i ($1 \leq i \leq N$), в които всеки трябва да играе с всеки следващ, като в S_i не включваме приятел i (защото i трябва да играе с всички от S_i). Като разгледаме приятел i в ден i, то множеството му S_i ще е точно с приятелите, с които трябва да играе. Ако m е приятелят с най-малък номер в S_i, то трябва просто да прехвърлим частта $S_i \setminus \{m\}$ към S_m. Важно е прехвърлянето да стане към най-малкия номер, за да спазваме поредността на дните/приятелите.

Сложност: $O(N^2 \log N)$.

Решение на трета подзадача – 55 точки (=20+30+5)

Можем отново да представяме множествата с булеви масиви или малко да оптимизираме горното решение като например трием старите множества S_i след преминаване към следващия ден. Така паметта ни от $O(N^2)$ става само $O(M)$ и това позволява кеша да работи по-ефективно, защото не се простираме в огромно количество памет.

Сложност: $O(N^2)$ или оптимизирано $O(N^2 \log N)$.

Решение на четвърта подзадача – 65 точки (=20+30+5+10)

Често при поддържане на множества можем да приложи оптимизация с *bitset*. Тук случаят е такъв, защото най-тежката ни операция е обединяване на множества, която е възможно да се реализира по-бързо като направим побитово ”или” на двата *bitset*-а, задаващи множествата. Допълнително метода `_Find_first` (извън стандарта) е удобен, за да намираме най-малкият номер в множеството.

Сложност: $O(\frac{N^2}{64})$.

Решение на пета подзадача – 15 точки (=0+0+0+0+15)

Възползваме се от допълнителното ограничение с това, че даден приятел ще бъде добавян най-много веднъж в множество. Така можем да поддържаме текущо множество отново със свойството, че трябва да има игра между всеки и всеки от следващите. Започваме с множество, в което включваме приятел 1, след което го махаме и добавяме неговите начални игри от първия ден, като това ще са игрите, в които той участва. След това се насочваме към приятеля с най-малък номер в множеството, премахваме го и добавяме неговите начални игри, така получаваме игрите с негово участие. Продължаваме по същия начин, докато не се изпразни множеството. Тази процедура трябва да я изпълним за всеки необработен приятел, като няма как да повтаряме приятели по време на процедурата заради свойството на подзадачата. Удобно е да се използва приоритетна опашка вместо *set*.

Сложност: $O(M \log N + M)$.

Пълно решение – 100 точки

По същество това е и същото решение за 50 точки, но използваме техниката *small-to-large* при обединяването на множествата. За незапознатите, идеята е, когато обединяваме две множества да добавяме елементите на множеството с по-малко елементи към това с повече елементи. Това позволява независимо какви обединения правим, да имаме амортизирана $O(\log N)$ сложност на обединяване или общо $O(N \log N)$. Анализът е много лесен - когато един елемент го добавяме към ново множество, то получаваме множество с поне два пъти повече елементи от старото му. Така от гледната точка на всеки един елемент сме го добавяли към друго множество най-много $\log N$ пъти.

Сложност: $O(N \log^2 N + M)$.

Тази задача беше малко по-трудна за общински кръг, но е дадена с идеята да е поучителна. Използват се стандартни оптимизации, които са полезни и в много други случаи и по-сериозни задачи.

Автор: Илиян Йорданов (zaeta)