АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА ТЪРГОВИЯ

Задачата се решава използвайки метода на динамичното оптимиране, комбиниран с обхождането в дълбочина.

Първото основно наблюдение е, че градовете и пътищата на страната Олимпия образуват дърво и R (столицата) е неговият корен.

Нека с weight[i] означим разликата от стойността на стойността на продажбава в i-тия връх и таксата, която трябва да се плати за да се в i-тия връх (ако има такава, иначе заместваме с 0).

Започваме обхождането от корена. Нека имаме три масива: best_u[], best_d[], best[], в които накрая на обхождането на даден връх целта ни е да имаме съответно: максимална печалба, ако в този връх има задължително търговец, който се е запътил „нагоре“, максимална печалба, ако в този връх задължително има търговец, който се е запътил „надолу“ и максимална печалба за поддървото с корен този връх, без да се интересуваме има или няма търговец в този връх. Ако допуснем, че имаме сметнати всички от гореспоменатите стойности за наследниците на текущо разглеждания връх, то можем да направим рекурентните сметки. Нека current е текущо разглеждания връх в обхождането в дълбочина и нека със sum означим сумата от най-добрите решения за всеки от наследниците му (т.е. сумата от best-овете на наследниците му).

За best_u[current] имаме два варианта – или в текущия връх по принцип да имаме търговец (и да решим той да се движи към корена), или такъв търговец да дойде от някой от наследниците му. В първия случай сметката е sum + weight[current] , а във втория е sum – best[i] + best_u[i] + weight[current], където i е някой от наследниците. Съответно, от гореизброените два случая трябва да вземем максималния. Първият случай е възможен само, ако имаме търговец в текущия връх, а втория случай е възможен за всяко i, за което е възможно да позиционираме търговец, който се движи нагоре. Ако първото условие не е изпълнено, както не е изпълнено и второто условие за нито едно i, то трябва да отбележим, че не е възможно да постигнем резултат best_u за current, тъй като е невъзможно да позиционираме търговец в current, който да се движи нагоре.

За best_d[current] не се интересуваме дали е възможно да стигнем отгоре с някакъв търговец (т.е. на етапа на смятане, ние няма как да знаем дали е възможно да „спуснем” търговец до current), в този смисъл това е по-лесния вариант, тъй като останалата част е аналогична като best_u. Имаме два варианта – или да приключим пътя на търговеца по-посока надолу в current (отново, не се интересуваме от къде идва той), или той да продъжи някъде надолу по някой от наследниците. В първия случай сумата е: weight[current] + sum, а във втория случай е sum – best[i] + best_d[i] + weight[current], където i е пряк наследник на current. Трябва, отново, да вземем максимума от тези суми.

За best[current] е най-лесния случай. Трябва да вземем максимума от 1. sum, 2. best_u[current], ако сме отбелязали като възможен този случай (т.е. да позиционираме някой търговец в current, който се е запътил нагоре) и 3. best_d[current], ако в current имаме поставен търговец по условие.

Отговорът на задачата стои в best[root], където root е коренът на дървото.

Автор: Момчил Иванов