1 Wprowadzenie

Zanim przystąpimy do przedstawienia rozwiązania zadania odnotujmy następujące obserwacje. Cała nasza wiedza na temat liczby pielgrzymów w danym momencie będzie się opierała na założeniu, że Jack nigdy nie miał do czynienia z niecałkowitymi liczbami euro. Prowadzi nas to do następujących obserwacji:

Twierdzenie 1

Załóżmy, że część danych ma następującą postać: **IN**/**OUT** a_1 **PAY** k_1 **PAY** k_2 ... **PAY** k_n **IN**/**OUT** a_2 oraz po tym jak a_1 pielgrzymów przyłączyło się (opuściło) do grupy, grupa liczyła dokładnie x osób. Z tego wynika, że x jest dzielnikiem $k_1 + k_2 + \ldots + k_n$. Ponadto dowolna wartość x o tej własności może być liczbą osób w grupie po wierszu **PAY** k_n.

Istotnie, jeśli x nie byłby dzielnikiem $k_1 + k_2 + \ldots + k_n$ po tym jak a_2 osób przyłączyło się bądź opuściło grupę Jack musiałby natrafić na niecałkowitą liczbę euro, co jak wiemy nigdy nie nastąpiło.

Obserwacja 2

Każdy wpis typu **COLLECT** nie wnosi żadnych informacji. Zatem usunięcie ich nie ma wpływu na szukaną odpowiedź.

Faktycznie, ponieważ dana jest tylko jedna strona notesu, Jack mógł zebrać wystarczającą kwotę tuż przed początkiem naszej strony.

Obserwacja 3

Początkowa sekwencja wpisów typu **PAY** nie wnosi żadnych informacji.

Załóżmy, że dane mają postać $i_1, i_2, ..., i_n$, gdzie $i_1, ..., i_k$ są typu **PAY**, i_{k+1} jest typu **IN** lub **OUT** oraz i_j nie jest typu **COLLECT** dla żadnego $1 \leq j \leq n$. Ponadto, załóżmy że x jest rozwiązaniem dla ciągu $i_{k+1}, i_{k+2}, \ldots, i_n$. Wtedy x jest również rozwiązaniem dla ciągu wpisów $i_1, i_2, ..., i_n$, gdyż poprzednia strona notatnika mogła kończyć się wpisami $j_1, j_2, \ldots, j_m$ typu **PAY**, tak aby x był dzielnikiem sumy posiadanej przez Jack'a.

Obserwacja 4

Jeśli dane kończą sie ciągiem wpisów typu **PAY** możemy te wpisy zignorować.

Istotnie, dowolna liczba euro jaką zapłacił na końcu Jack nie ma wpływu na niecałkowitą liczbę euro, gdyż nikt nie przyłączył się ani nie opuścił grupy od tego momentu.

Obserwacja 5

Jeśli dany ciąg wpisów nie zawiera informacji typu **PAY** (zawiera jedynie wpisy typu **IN** oraz **OUT**) oraz $\Delta p_1, \Delta p_2, \ldots, \Delta p_n$ jest ciągiem zmian liczby osób w grupie (Δp_i odpowiada i-temu wierszowi: **IN** Δp_i lub **OUT** $-\Delta p_i$), wtedy dowolna liczba osób nie mniejsza niż $min_k - (\Delta p_1 + \Delta p_2 + \ldots + \Delta p_k)$ może być początkową liczbą pielgrzymów.

Mając na uwadze wszystkie powyższe obserwacje oraz twierdzenie możemy przedstawić następujący schemat rozwiązania:

1. Usuwamy wszystkie wpisy **COLLECT**.
2. Usuwamy wszystkie początkowe oraz końcowe wpisy typu **PAY** (jeśli takie występują).
3. Jeśli (po wykonaniu pierwszych 2 kroków) nie występują żadne wpisy typu **PAY** obliczamy minimalną liczbę pielgrzymów na podstawie obserwacji 5.

4. W przeciwnym wypadku mamy do czynienia z ciągiem składającym się z wpisów **IN/OUT** oraz **PAY**.

5. Weźmy pierwszą sekwencję informacji **PAY**, obliczmy ich sumę S.

6. Dla każdego dzielnika liczby S sprawdźmy, czy dzielnik ten może być liczbą osób w grupie w danym momencie, biorąc pod uwagę pozostałe wpisy.

Należy zaznaczyć, że powinniśmy znaleźć wszystkie dzielniki liczby S w czasie $O(\sqrt{S})$.

Ostatnią rzeczą jaką musimy umieć zrobić jest sprawdzenie, czy dana liczba m może stanowić liczność grupy biorąc pod uwagę pozostałe wpisy. Możemy tego dokonać wykonując prostą pętlę po wszystkich wpisach. Będziemy uaktualniać liczbę osób w grupie po każdym wpisie typu **IN/OUT** oraz dla każdego bloku informacji **PAY** będziemy obliczać ich sumę i sprawdzać czy jest ona podzielna przez aktualną liczbę osób w grupie (jeśli nie jest podzielna, to początkowa liczba osób w grupie była nieprawidłowa).

2 Rozwiązanie wzorcowe

Rozwiązanie wzorcowe (plik *pil.{cpp/java}*) jest implementacją powyższego algorytmu. Na początku usuwamy zbędne wpisy (opisane w punkcie 1) oraz 2)). Później wywołanie funkcji *no_pays()* obsługuję przypadek, gdy nie pozostał ani jeden wpis typu **PAY**. Następnie wywołana jest główna funkcja programu. Sumowane są tam wartości w pierwszym bloku operacji **PAY** (nazwijmy tą wartość sum), oraz dla każdej pary jej dzielników j oraz sum/j dodajemy je do wyniku, jeśli są to możliwe liczby osób w danym momencie w grupie. Odbywa się to za pomocą wywołania funkcji *possible()*, sprawdzając wcześniej, czy rozważana wartość jest nie mniejsza niż $1+$ zmiana liczby osób w grupie spowodowana wpisem **IN/OUT**. Funkcja *possible()* implementuje schemat przedstawiony na końcu poprzedniej sekcji.

Niech N będzie rozmiarem danych. Złożoność obliczeniowa rozwiązania wzorcowego wynosi $O(N\sqrt{2000N})$, natomiast pamięciowa to $O(N)$.

3 Rozwiązanie wolne 1

Rozwiązanie wolne 1 (*pils1.{cpp/java}*) jest bardzo podobne do rozwiązania wzorcowego. Jedyną różnicą jest sposób znajdowania wszystkich dzielników liczby S poprzez pętlę po wszystkich liczbach naturalnych nie większych niż S, co zajmuje istotnie więcej czasu. Niech N będzie rozmiarem danych. Złożoność obliczeniowa wynosi $O(2000N^2)$, natomiast pamięciowa $O(N)$

4 Testy

Testy umieszczone są w następujących plikach:

- *pil0.in* — test przykładowy,

- *pil1.in* — prosty test poprawnościowy,
- *pil2.in* — bardziej złożony test poprawnościowy,
- *pil3.in* — test wydajnościowy składający się z krótkich (1-3 elementowych) początkowych i końcowych bloków **IN/OUT** oraz dużego (40-48 elementowych) bloku wpisów typu **PAY** ze stosunkowo dużymi kwotami (1850-2000). Zatem sumaryczna kwota jest wystarczająco duża aby odróżnić rozwiązanie wzorcowe od rozwiązania wolniejszego.
- *pil4.in* — losowy test wydajnościowy, składa się od z naprzemiennych bloków **IN/OUT** (najczęściej krótkich) oraz **PAY**. Pierwszy znaczący blok wpisów **PAY** jest stosunkowo długi, tak aby wszystkie dzielniki, które musimy znaleźć występowały w liczbie $50,000$-$70,000$.

Pierwsze dwa testy zostały stworzone ręcznie, pozostałe zostały wygenerowane w sposób automatyczny przez program *pilingen.cpp*, gdyż testy efektywnościowe powinny zawierać około $10,000$ przypadków testowych, co daje łączną liczbę $400,000$-$500,000$ wierszy. Ponieważ informacje **COLLECT** nie są istotne dla rozwiązania zadania, tylko kilka takich wpisów znajduje się w losowych miejscach ostatnich dwóch testów.