Plac Budowy

Dawid Sroczyk

Firma budowlana Grafexpol jest odpowiedzialna za budowę budynków mieszkalnych na nowo powstającym osiedlu w Warszawie. Po intensywnej pracy przez cały rok, firmie udało się zrealizować budowę trzech nowoczesnych budynków, które zostały połączone w taki sposób, że tworzą literę „U”.
Niestety, niedoświadczeni pracownicy nie usunęli maszyn z placu budowy przed końcem zimy, a po wiosennych roztopach grunt stał się miękki i mokry, co znacznie utrudnia usunięcie ciężkiego sprzętu.
Plac budowy ma kształt prostokąta o szerokości `w` i wysokości `h`, a powierzchnia jest podzielona na kwadratowe pola. Każda maszyna, wyjeżdżając z danego pola, zostawia w jego ziemi głębokie bruzdy, przez co każde pole ma określoną liczbę, która wskazuje, ile razy może z niego wyjechać maszyna.
Pomóż firmie opracować system, który zminimalizuje jej straty i umożliwi bezpieczne wyprowadzenie jak największej liczby maszyn z placu budowy. Pozostałe maszyny staną się atrakcją turystyczną, ale również prawdziwą zmorą dla mieszkańców osiedla.
Formalnie:

- szerokość placu budowy to `w`, a wysokość to `h`
- dla każdego pola (`i`, `j`) na placu, wartość `P[i,j]` oznacza ile maszyn może łącznie wyjechać z tego pola
- Maszynę można przesunąć w jednym ruchu o jedno pole w górę, jedno pole w dół, jedno pole w lewo albo jedno pole w prawo
- Maszynę uważamy jako wyprowadzoną z placu budowy, jeśli wykonując serię ruchów znajdzie się w zerowym wierszu i następnie wykona ruch w górę

Etap I(1.5p)

W etapie pierwszym, celem jest wyprowadzenie z placu budowy jak największej liczby maszyn.
Dane:

- `int[,] P` - tablica wymiaru `h` $\times$ `w`, zdefiniowana tak, jak w treści zadania
- `(int row, int col)[] MachinePos` - tablica zawierająca informację o początkowym ustawieniu maszyn. Wartość `MachinePos[i]` oznacza pozycję maszyny o indeksie `i`

Wynik:

- `int savedNum` - maksymalna liczba maszyn, którą można wyprowadzić z placu
- `int[] Saved` - tablica zawierająca w dowolnej kolejności indeksy `s` wyprowadzonych maszyn. Jeśli istnieje wiele rozwiązań, zwrócić dowolne z nich.

Etap II(1p)

Po wyciągnięciu maszyn z kilku placów budowy firma zauważyła, że najcenniejsze maszyny trzymane są zazwyczaj w głębi placu, a najstarsze i najmniej wartościowe znajdują się przy samym brzegu. Z tego powodu najczęściej ratowane były te tańsze maszyny. Co więcej, przesunięcie maszyny o jedno pole zajmuje dużo czasu, przez co spora część wypłaty pracowników przeznaczana jest na przesuwanie maszyn. Nowa polityka firmy polega na maksymalizowaniu różnicy między sumaryczną wartością uratowanych maszyn a sumarycznym kosztem ich przesuwania.
Dane:

- `int[,] P` - tablica zdefiniowana tak samo jak w etapie I
- `(int row, int col)[] MachinePos` - tablica zdefiniowana tak samo jak w etapie I
- `int[] MachineValue` - tablica zawierająca informację o wartości maszyn. Wartość `MachineValue[i]` oznacza wartość maszyny o indeksie `i`.

- `int moveCost` - koszt przesunięcia maszyny o jedno pole

Wynik:

- `int bestProfit` - maksymalna różnica między sumaryczną wartością wyprowadzonych maszyn, a kosztem ich wyprowadzenia
- `int[] Saved` - tablica zawierająca w dowolnej kolejności indeksy maszyn, których wyprowadzenie maksymalizuje zysk z całej operacji. Jeśli istnieje wiele rozwiązań, zwrócić dowolne z nich

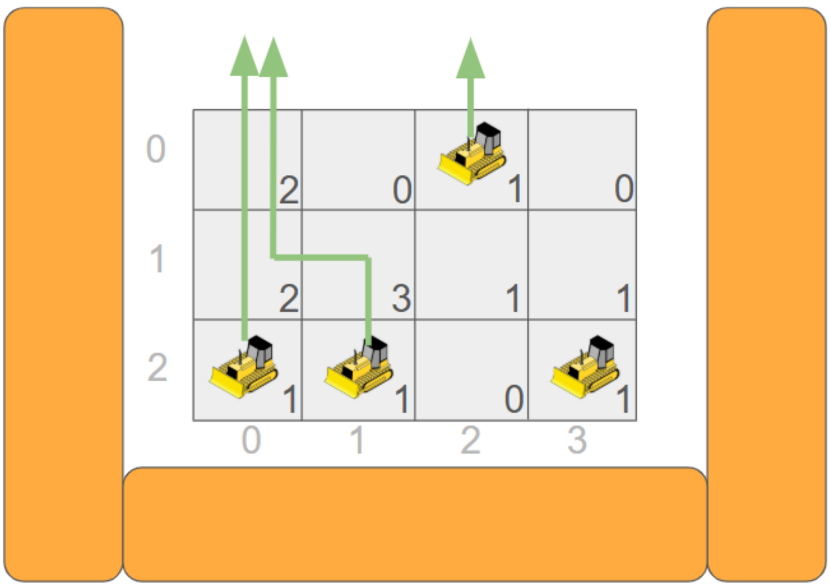
Uwagi

- W ustawieniu początkowym, na każdym polu stoi co najwyżej jedna maszyna
- W wyniku przesuwań, na jednym polu może stać wiele maszyn
- Maszyna niszczy pole dopiero przy wyjeżdżaniu z niego - nie podczas wjeżdżania i nie podczas stania na nim. Przykładowo, jeśli `P[i,j] = 3`, to na pole `(i,j)` może wjechać dowolnie duża liczba maszyn, ale wyjechać będą mogły tylko 3
- tablica `P` może przyjmować wartości dodatnie i zerowe, ale nie może ujemnych
- W etapie II, koszt przesunięcia każdej maszyny o jedno pole jest taki sam
- W etapie II, korzystnym może być uratowanie mniejszej liczby maszyn, ale bardziej cennych. W szczególnym przypadku, najbardziej może opłacać się nieratowanie żadnej maszyny
- W etapie I oczekiwana złożoność wynosi $O(\mathtt{w} \cdot \mathtt{h} \cdot \mathtt{savedNum})$
- W etapie II oczekiwana złożoność wynosi $O(MC)$, gdzie MC to złożoność algorytmu MinCost-MaxFlow dla grafu z $O(\mathtt{w} \cdot \mathtt{h})$ wierzchołkami i $O(\mathtt{w} \cdot \mathtt{h})$ krawędziami

Przykłady

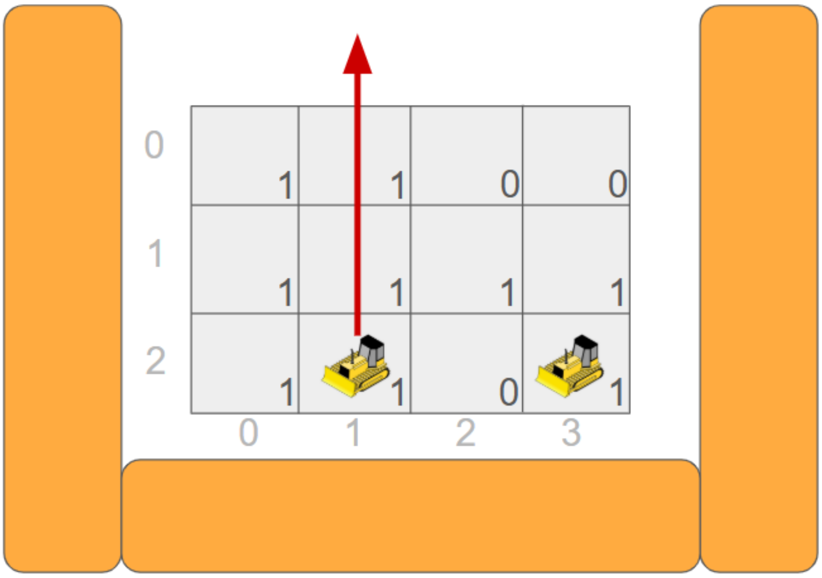
Etap I, przykład 1

Przykład poprawnego wyprowadzenia trzech maszyn. Liczba w rogu pola `(i,j)` oznacza wartość `P[i,j]`. Widzimy, że maszyna na polu `(2, 3)` nie może już zostać uratowana, ponieważ uratowanie pozostałych maszyn, uczyniło niektóre pola nieprzejezdnymi.

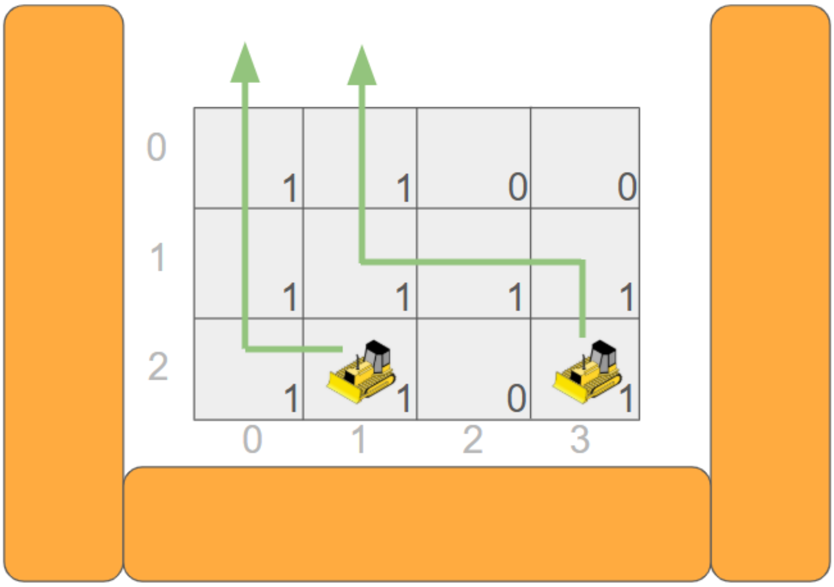


Etap I, przykład 2

Jeśli zdecydujemy się na takie wyprowadzenie, to uratujemy tylko jedną maszynę



Ale jeśli rozplanujemy inaczej, to uratujemy obydwie



Etap II, przykład

Ustalmy, że wartość maszyny niebieskiej to 1000\$, żółtej 400\$, a `k` $=$ 100\$. Wówczas najbardziej opłaca nam się wyprowadzić jedynie niebieską maszynę. Koszt wyprowadzenia żółtej to co najmniej 500\$, więc nie opłaca się jej ratować. Przy uratowaniu jedynie niebieskiej, szukana różnica wynosi $1000\$ - 3 \cdot 100\$ = 700\$$