Opracowanie: AQU
Aquapark

1 Rozwiązanie I (brutalne)

Najprostsze możliwe rozwiązanie polega na przejrzeniu kolejno dla każdego ratownika wszystkich pól na planszy. Jeżeli odległość liczona w sposób opisany w treści zadania (czyli w tzw. metryce miejskiej lub metryce Manhattan) tego pola od miejsca, w którym stoi ratownik, jest dostatecznie mała, to dodajemy rozważane pole do wyniku.

Przykładowy pseudokod:

```
function odleglosc(x1, y1, x2, y2)
begin
    return abs(x1 - x2) + abs(y1 - y2);
end;

wczytaj(n, r);
wczytaj_opis_aquaparku(a[][]);

for ratownik := 1 to r do
begin
    wczytaj_opis_ratownika(x, y, l);
    wynik := 0;
    { dla każdego segmentu aquaparku }
    for i := 1 to n do
        for j := 1 to n do
            { jeżeli segment jest w zasięgu ratownika }
            if odleglosc(x, y, i, j) <= l then
                wynik := wynik + a[i][j];
    writeln(wynik);
end;
```

Ponieważ dla każdego ratownika przeglądamy n^2 pól, złożoność czasowa rozwiązania wynosi $O(r \cdot n^2)$. Rozwiązanie to otrzymywało 25% punktów.

2 Rozwiązanie II

W połowie testów każdy basenik jest chroniony przez co najwyżej jednego ratownika. Korzystając z tego, możemy znacznie ulepszyć rozwiązanie, dla każdego ratownika przeszukując tylko pola, które są w jego zasięgu.

```
wczytaj(n, r);
wczytaj_opis_aquaparku(a[][]);

for ratownik := 1 to r do
begin
    wczytaj(x, y, l);
    wynik := 0;
    poczX := max(1, x - l);
    koniecX := min(n, x + l);
    for i := poczX to koniecX do
    begin
        { ile maksymalnie ruchów można jeszcze wykonać w pionie }
        szer := l - abs(x - i);
        poczY := max(1, y - szer);
        koniecY := min(n, y + szer);

        { przeglądamy wszystkie pola w tej kolumnie, do których }
        { ratownik ma dostatecznie blisko }
        for j := poczY to koniecY do
                wynik := wynik + a[i][j];
    end
    wypisz(wynik);
end
```

W połowie przypadków takie ulepszone rozwiązanie przejrzy co najwyżej $O(n^2)$ pól, więc jego złożoność czasowa wyniesie dla nich $O(r+n^2)$. Za to rozwiązanie przewidziane było 50% punktów, zgodnie z informacją w treści zadania.

3 Rozwiązanie III

Następną optymalizacją może być przyspieszenie sumowania liczb a_{ij} w obszarach będących w zasięgu ratowników.

W tym celu możemy po wczytaniu „mapy" aquaparku dodać do każdego pola sumę wszystkich liczb leżących w jego kolumnie powyżej niego. Wtedy sumowanie liczby dzieci w zasięgu ratownika będzie wymagało tylko $O(n)$ kroków — tyle, ile jest kolumn, które zawierają przynajmniej jedno pole w zasięgu ratownika.

```
wczytaj(n, r);
wczytaj_opis_aquaparku(a[][]);

for i := 1 to n do
begin
    d[i][0] := 0;
    d[0][i] := 0;
end;
{ obliczamy łączną liczbę dzieci w kolumnie i, w wierszach od 1 do j włącznie }
for i := 1 to n do
    for j := 1 to n do
        d[i][j] := d[i][j - 1] + a[i][j];

for ratownik := 1 to r do
begin
    wczytaj(x, y, l);
    wynik := 0;
    poczX := max(1, x - l);
    koniecX := min(n, x + l);
    for i := poczX to koniecX do
    begin
        { ile maksymalnie ruchów można jeszcze wykonać w pionie }
        szer := l - abs(x - i);
        poczY := max(1, y - szer);
        koniecY := min(n, y + szer);
        { dodaję do wyniku pola w kolumnie i, w wierszach od poczY do koniecY włącznie }
        wynik := wynik + d[i][koniecY] - d[i][poczY - 1];
    end;
    wypisz(wynik);
end;
```

Ponieważ dla każdego ratownika wykonujemy liczbę kroków rzędu n, złożoność czasowa tego rozwiązania wynosi $O(n \cdot r)$. Liczba punktów przewidziana za nie to 70%.

4 Rozwiązanie IV (wzorcowe)

Ostatnią potrzebną optymalizacją jest zredukowanie czasu liczenia sumy w zasięgu ratownika do stałej liczby operacji.

Spróbujmy rozszerzyć pomysł z poprzedniego rozwiązania. Umiemy już liczyć w czasie stałym sumy w odcinkach kolumn po wcześniejszym przesumowaniu pewnych pól. Jeżeli jednak na początku wykonamy taką operację sumowania nie raz, ale dwa razy — zarówno w pionie jak i w poziomie — to będziemy w stanie w czasie stałym liczyć sumy w całych prostokątach!

Przyjrzyjmy się temu pomysłowi dokładniej. W poprzednim rozwiązaniu obliczaliśmy tablicę d, która w komórce (i, j) zawierała sumę liczb z tablicy a ze wszystkich komórek w kolumnie i znajdujących się ponad nią:

$$d[i][j] = \sum_{k \leqslant j} a[i][k].$$

Teraz spróbujmy policzyć tablicę d, która w każdej komórce będzie miała sumę wszystkich, które leżą na lewo i ponad tą komórką w tablicy a:

$$d[i][j] = \sum_{k \leqslant j} \sum_{l \leqslant i} a[l][k].$$

Możemy to wykonać podobnie jak poprzednio, w sposób dynamiczny, jednak tym razem posługując się następującym wzorem:

$$d[i][j] = d[i][j-1] + d[i-1][j] - d[i-1][j-1] + a[i][j].$$

Teraz jeżeli chcielibyśmy szybko poznać sumę liczb w tablicy a w prostokącie $(x1, y1, x2, y2)$ (gdzie $x1 \leqslant x2$ oraz $y1 \leqslant y2$), to będzie ona równa:

$$d[x2][y2] - d[x2][y1-1] - d[x1-1][y2] + d[x1-1][y1-1].$$

Hmm... No więc świetnie — umiemy policzyć sumę w każdym prostokącie w czasie stałym. Ale do czego to właściwie może się nam przydać? Przecież w zadaniu potrzebujemy liczyć szybko sumy w obszarach leżących w zasięgu ratowników, a one w ogólności nie są prostokątami!

Zaraz, ale czy na pewno? Przypatrzmy się bliżej... Tak naprawdę są one w przybliżeniu prostokątami, tylko o bokach nierównoległych do boków aquaparku, ale za to równoległych do jego przekątnych. Możemy to spostrzeżenie sprytnie wykorzystać. Jeżeli obrócimy całą mapę aquaparku o kąt 45°, to obszary chronione przez ratowników stają się prostokątami, dla których możemy już wykonać powyższą sztuczkę.

Obracanie tych pól możemy wykonać już przy wczytywaniu danych. Należy przy tym pamiętać, że rozmiar tablicy d zwiększy się w obu wymiarach dwukrotnie. Aby obrócić mapę o 45°, należy zamienić współrzędne (i, j) na $(j+i, j-i)$ — kto nie wierzy, niech sprawdzi na kilku przykładach. Aby jednak otrzymać współrzędne z zakresu $[1, 2 \cdot n - 1]$, powinniśmy przesunąć je jeszcze nieznacznie (czyli dodać do nich pewne stałe). Tak więc ostatecznie będziemy zamieniać współrzędne (i, j) na $(j + i - 1, j - i + n)$.

Oto pseudokod rozwiązania wzorcowego:

```
{ oblicza sumę liczb w prostokącie [x1, x2] x [y1, y2] tablicy a[][] }
function oblicz_sume_prostokata(x1, y1, x2, y2)
begin
    x1 := max(0, x1);
    y1 := max(0, y1);
    x2 := min(2 * n, x2);
    y2 := min(2 * n, y2);
    return d[x2][y2] - d[x2][y1 - 1] - d[x1 - 1][y2] + d[x1 - 1][y1 - 1];
end

wczytaj(n,r);
{ wczytywanie mapy aquaparku i obracanie jej }
for i := 1 to n do
    for j := 1 to n do
        begin
            wczytaj(tmp);
            { obrót planszy o 45 stopni }
            x := j + i - 1;
            y := j - i + n;
            a[x][y] := tmp;
        end;

{ dynamicznie obliczamy tablicę d[i][j] }
for i := 1 to n do
begin
    d[i][0] := 0;
    d[0][i] := 0;
end;
for i := 1 to 2 * n do
    for j := 1 to 2 * n do
        d[i][j] := d[i][j - 1] + d[i - 1][j] - d[i - 1][j - 1] + a[i][j];

for ratownik := 1 to r do
begin
    wczytaj(c, d, l);
    x := c + d - 1;
    y := n - c + d;
    wynik := oblicz_sume_prostokata(x - l, y - l, x + l, y + l);
    wypisz(wynik);
end;
```

Złożoność rozwiązania wzorcowego wynosi $O(n^2 + r)$ we wszystkich przypadkach testowych, ponieważ wczytanie mapy, obrócenie jej oraz obliczenie sum częściowych wymaga $O(n^2)$ kroków, zaś dla każdego ratownika potrzebujemy już tylko stałej liczby operacji.