SmartPark

CIN UFPE

✓

Time Limit
3 seconds

Memory Limit
768 MB

Output Limit
4 MB

Titulo

SmartPark

Descrição

Num planeta distante muito povoado, os habitantes deslocam-se em veículos autônomos com placas de matrícula formadas por 6 letras letras no conjunto `A` a `Z`. Os parques de estacionamento de locais públicos passaram a funcionar de forma modular da seguinte maneira:

1. O estacionamento possui `M` vagas, numeradas de `0` a `M−1`. Inicialmente, está disponível a quantidade mínima `M=Mmin` de vagas.
2. Ao chegar no pórtico de entrada, o veículo de placa `P=P[0]...P[5]` é dirigido a uma vaga com base numa chave `K(P)` associada à sua placa, calculada como

```
K(P) = P[0]*26^0 + P[1]*26^1 + ... + P[5]*26^5
```

Nesse cálculo, a letra `A` equivale ao valor `0`, a letra `B` ao valor `2`, e assim sucessivamente até o valor `Z=25`.

3. O número da vaga inicialmente atribuída ao veículo de chave `K` é calculada como

```
H(K) = K mod M.
```

4. Ao chegar nessa vaga, caso ela já esteja ocupada, o veículo deve dirigir-se às vagas `(H(K)+1) mod M`, `(H(K)+2) mod M`, `(H(K)+3) mod M`, e assim sucessivamente, até encontrar uma vaga livre.

Para garantir que sempre haja vagas livres, o sistema monitora a quantidade de vagas ocupadas `N` e, ao chegar um novo veículo, **antes** que ele seja admitido, o sistema verifica se a **taxa de ocupação** `F=N/M` é superior a um valor máximo limite `Fmax < 1.0`. Caso `F > Fmax`, um novo módulo com a mesma quantidade de vagas atualmente disponíveis (ocupadas ou não) é aberto. Portanto, passamos a ter `2M` vagas numeradas de `0` a `2M−1`. **Todos** os veículos já estacionados são então automaticamente retirados das suas vagas, que passam a ser consideradas todas livres novamente, e redistribuídos pelas novas vagas, um de cada vez, pela ordem das suas antigas vagas (primeiro o veículo da vaga 0, se houver, depois da 1, etc.), seguindo o mesmo processo 2-4 acima. Só após isso, o novo veículo é admitido e estacionado segundo o mesmo processo.

Quando um veículo sai do estacionamento, a vaga por ele ocupada fica vazia porém ainda marcada como **indisponível**, conforme indicação de um aviso luminoso. Entretanto, **imediatamente após** cada veículo deixar o estacionamento, o sistema verifica se a **taxa de ocupação efetiva** `E=N'/M`, onde `N'` é a quantidade de vagas efetivamente ocupadas por veículos, é inferior a uma taxa mínima `Fmin < Fmax/2`. Se `E < Fmin`, então a metade das vagas é fechada, passando-se a `M/2` vagas, exceto se já estivermos com a quantidade mínima de vagas `Mmin` (ou seja, `M = MAX(M/2, Mmin)`). **Todos** os veículos são então redistribuídos seguindo o mesmo processo de redistribuição descrito acima. Repare que, imediatamente antes da redistribuição, todas as vagas são liberadas e, após a redistribuição, haverá apenas vagas efetivamente ocupadas ou livres, e mais nenhuma apenas vazia porém sinalizada como indisponível. Caso `E >= Fmin`, não há redistribuição. Nesse caso, as vagas vazias marcadas como ocupadas contarão como vagas ocupadas para efeito do cálculo da taxa de ocupação `F=N/M` que ocorre na chegada de novos veículos. Caso haja alocação de novas vagas e redistribuição por ocasião da chegada de um novo veículo, essas vagas apenas marcadas como indisponíveis serão também efetivamente liberadas.

Ao voltar para procurar o seu veículo, o proprietário pode encontrar o estacionamento numa configuração diferente e, portanto, é necessário seguir um processo análogo aos passos 2-4 acima, ou seja, recalcular `H(K)` com base no valor de corrente de `M` (número total de vagas) e dirigir-se inicialmente a essa vaga. Caso encontre lá o seu veículo, ótimo. Senão, será necessário procurar nas vagas consecutivas `(H(K)+1) mod M`, `(H(K)+2) mod M`, `(H(K)+3) mod M`... até encontrá-lo. Uma busca pode ser dada como falha quando, e apenas quando, encontramos a primeira vaga **efetivamente vazia** sem encontrar o veículo. Caso a vaga esteja apenas sinalizada como indisponível, a busca deve continuar.

Entretanto, após alguns meses de funcionamento, alguns clientes passaram a reclamar que chegam a passar muito tempo procurando seus veículos. Chamemos de **distância de busca** o número de vagas verificadas até encontrar um veículo ou concluir que ele não se encontra no estacionamento além da vaga original. No melhor caso a distância é `0` (veículo estacionado na vaga original) e, no pior caso, a distância pode corresponder a `N−1`, onde `N` é o número de vagas ocupadas. A empresa administradora desenvolveu então uma atualização do sistema, no qual o veículo estacionado mantém um registro da distância percorrida a partir da posição original. Mais especificamente, para estacionar um veículo de chave `K`:

1. A posição da vaga original `H(K)=K mod M` é calculada da mesma forma

2. O veículo tenta estacionar na vaga `H(K)` com distância `D=0`.

3. Caso a vaga original esteja ocupada, o veículo verifica as posições subsequentes `(H(K)+D) mod M` com distâncias `D=1,2,...`

4. Se a posição `J=(H(K)+D) mod M` estiver livre, o veículo é estacionado nessa vaga com distância `D`

5. Se a posição `J=(H(K)+D) mod M` estiver ocupada por um veículo de chave `K'` com distância `D'`, temos:

   i. Se `D <= D'`, então o veículo `K` ainda não está mais afastado do que `K'` da sua posição original. Nesse caso, a busca continua com o veículo `K` para as posições seguintes com distâncias `D+1, D+2, ...`

   ii. Porém, se `D > D'` então isso significa que o veículo `K` já está mais afastado da sua posição original do que o veículo `K'`. Nesse caso, o veículo `K` "rouba" a vaga `J` e é colocado ali a uma distância `D`. O veículo `K'` é forçado continuar a busca por uma nova vaga daquela posição em diante `(J+1) mod M, (J+2) mod M, ...` a distâncias `D'+1, D'+2,...` respeitando esse mesmo procedimento (passos 4-5).

O aumento/diminuição/redistribuição de vagas são avaliados da mesma forma da primeira versão. A busca por um veículo com chave `K` nesta segunda versão é similar em tudo a da primeira versão, exceto por um critério extra. A busca começa na posição inicial `H(K)` e, no caso do veículo não estar estacionado nessa posição, continua nas posições seguintes `(H(K)+D) mod M` para `D=1,2,...`. Contudo, suponha que cheguemos a uma posição `J = (H(K)+D) mod M` e que lá encontremos um veículo `K'` com distância `D'`. Se `D' < D` então saberemos que o veículo `K` não pode estar no estacionamento pois, se estivesse, ele teria "roubado" essa vaga `J` e lá estaria com distância `D`. Nesse caso, a busca já deve ser encerrada (com distância `D`). Repare que as buscas, em ambas as versões do estacionamento, sempre ignoram as vagas sinalizadas indisponíveis (de onde saíram veículos estacionados) e passam direto por elas apenas incrementando a distância `D`.

A empresa deseja fazer simulações para estimar as diferenças entre as duas versões do sistema.

Input Specification

A entrada inicia por uma linha

```
Mmin Fmin Fmax
```

onde

- `Mmin` é um inteiro que indica a quantidade mínima/inicial de vagas
- `Fmin` é um inteiro que indica a taxa de ocupação **efetiva** mínima para efeito de redução de vagas **em pontos percentuais**
- `Fmax` é um inteiro indica a taxa de ocupação máxima para efeito de aumento de vagas **em pontos percentuais**

Seguem-se várias linhas, cada uma com um comando numa das formas

- `IN P` : chegada de um novo veículo com placa `P` (String de 5 letras maiúsculas)
- `OUT P` : saída de um veículo de placa `P`
- `SCH P` : procura um veículo de placa `P` no estacionamento

a entrada termina com uma linha

```
END
```

Notes

Os valores de `Fmin` e `Fmax` são dados em pontos percentuais. Assim, temos

```
0 < Fmin < Fmax/2 < Fmax < 100
```

Ou seja, um valor `Fmax=75` indica que a taxa de ocupação não pode ser superior a `75% = 0.75`.

Para evitar problemas de precisão com ponto flutuante, o teste se o estacionamento com `N` vagas ocupadas/indisponíveis das `M` vagas totais excede a taxa máxima de ocupação deve ser veito verificando se

```
100*N > M*Fmax
```

Da mesma forma, para verificar se a ocupação **efetiva** é inferior à taxa mínima deve-se testar se

```
100*N' < M*Fmin
```

onde `N'` é o número de vagas efetivamente ocupadas por um veículo.

Output Specification

Saída

O programa deve simular simultaneamente dois etacionamentos nas duas versões descritas acima.

Para cada comando `IN P` o programa deve incluir o novo carro no estacionamento e imprimir uma linha

```
J0 D0 J1 D1
```

onde

- `J0, J1` indicam as posições em que o veículo foi estacionado nas duas versões do estacionamento
- `D0, D1` indicam as distâncias em relação às posições originais nos dois estacionamentos

Para cada comando `OUT P` ou `SCH P` o programa deve imprimir uma linha

```
J0 D0 J1 D1
```

onde

- `J0, J1` indicam as posições em que o veículo estava estacionado nas duas versões do estacionamento
- `D0, D1` indicam as distâncias em relação às posições originais em que o veículo estava nos dois estacionamentos

Caso o veículo não esteja no estacionamento, deve ser impressa uma linha

```
 -1 -1 -1 -1
```

Ao final deve ser impressa uma linha com dois inteiros

```
Dmax0 Dmax1
```

correspondentes às distâncias máximas resultantes dentre todas as buscas (comandos `SCH`) nas duas versões do estacionamento.

Sample Input #1

```
200 10 75
IN TRPITD
IN IIPYBL
IN PXNXEH
IN PXULVI
IN HWYFTP
```

Sample Output #1

```
81 0 81 0
92 0 92 0
185 0 185 0
173 0 173 0
67 0 67 0
184 0 184 0
```

Sample Input #2

```
200 10 50
IN VKFTPK
IN IRPWUG
IN HDCUFY
IN RLKUKY
IN NPAFOK
```

Sample Output #2

```
5 0 5 0
38 0 38 0
61 0 61 0
167 0 167 0
107 0 107 0
192 0 192 0
```

Sample Input #3

```
200 10 50
IN CDEVPS
IN IDPKOK
IN JMEHYI
IN BOLEML
IN JQXMWN
```

Sample Output #3

```
88 0 88 0
10 0 10 0
89 0 89 0
153 0 153 0
45 0 45 0
46 0 46 0
```

Sample Input #4

```
10 20 95
IN WYCCXU
IN IKJYXU
IN PCMDKM
IN WLHQVC
IN VNPMZN
```

Sample Output #4

```
8 0 8 0
4 0 4 0
9 0 9 0
5 1 5 1
0 1 0 1
1 3 9 1
```

Sample Input #5

```
100 40 90
IN GTUIZS
IN VTEBPQ
IN ORRFFZ
IN DMRJBO
IN QORWON
```

Sample Output #5

```
96 0 96 0
51 0 51 0
8 0 8 0
31 0 31 0
97 1 97 1
59 0 59 0
```