**Решение за 5 точки**

Стандартно $Dynamic programming$ с битова маска.

Сложност: $O(2^{n}\*n)$

**Решение за 25 точки**

Строим следния граф:

$source\rightarrow 2\*i-1$ с $flow=1, cost= 0$

$2\*i-1\rightarrow 2\*i$ с $flow=1, cost= -infinity$

$2\*i\rightarrow sink$ с $flow=1, cost= 0$

$2\*i\rightarrow 2\*j-1$ с $flow=1, cost=|r\_{i}-l\_{j}|$ за $1\leq i<j\leq n$

И $infinity$ е достатъчно голямо число (например $5\*10^{13}$)

Отговорът на задачата е $n\*infinity+Minimum-cost flow problem $за построения граф за $flow=k$.

Имаме $N=O(n), M=O(n^{2})$, можем да приложим $Bellman–Ford algorithm$ за $O\left(N\*M\right)$ за намиране на минималното разстояние на всяка итерация.

Сложност: $O\left(k\*N\*M\right)=O\left(k\*n^{3}\right)$

**Решение за 35 точки**

За намиране минималното разстояние на всяка итерация можем вместо това да приложим:

1. $Shortest Path Faster Algorithm (SPFA)$
2. $Bellman–Ford algorithm$ с $break$, ако на поредното минаване през ребрата не е имало промяна в разстоянията.

**Решение за 50 точки**

Ще построим друг граф с повече върхове, но по-малко ребра:

Ще приложим $divide\\_and\\_conquer(l, r)$, остава да построим ребрата $l\leq i\leq av=\left⌊\frac{l+r}{2}\right⌋<j\leq r$. Остава да разкрием дали $\left|r\_{i}-l\_{j}\right|=+ r\_{i}-l\_{j} или- r\_{i}+l\_{j}$. Сортираме $r\_{i}$ и $l\_{j}$ в нарастващ ред, при фиксирано $r\_{i}$, $\left|r\_{i}-l\_{j}\right|=+ r\_{i}-l\_{j} $за някой префикс и $\left|r\_{i}-l\_{j}\right|=- r\_{i}+l\_{j}$ за оставащия суфикс, т.е. можем да построим оригиналните ребра за $r-l+1$ допълнителни върха и $3\*\left(r-l+1\right)$ допълнителни ребра.

При достатъчно малко $r-l$ може да се използва диркетно квадратното строене, което тогава всъщност строи по-малко ребра.

Общо получаваме $N=O(n\*log\_{2}(n)), M=O(n\*log\_{2}(n))$ и остава да приложим един от алгоритмите за минималното разстояние на всяка итерация от решението за 35 точки.

**Решение за 70 точки**

Остава да оптимизираме и самото пресмятане на потока с т.нар. “потенциали”, които са описани тук <https://codeforces.com/blog/entry/95823>

Основната идея е чрез тях да можем да прилагаме $Dijkstra's algorithm$ върху модифицирания граф, като в началото прилагаме веднъж $Bellman–Ford algorithm$

Сложност: $O\left(N\*M+k\*M\*log\_{2}(N)\right)=O\left(n^{2}\*log\_{2}\left(n\right)^{2}+k\*n\*log\_{2}\left(n\right)^{2}\right)$

**Решение за 100 точки**

Тъй като началният граф е ацикличен, можем да пресметнем началните потенциали вместо това с $Dynamic programming$, както е описано тук [https://codeforces.com/blog/entry/95823?#comment-891703](https://codeforces.com/blog/entry/95823#comment-891703)

Сложност: $O\left(k\*M\*log\_{2}(N)\right)=O\left(k\*n\*log\_{2}\left(n\right)^{2}\right)$

 Автор: Мартин Копчев