

Problem 2093: Minimum Cost to Reach City With Discounts

Problem Information

Difficulty: Medium

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

A series of highways connect

n

cities numbered from

0

to

$n - 1$

. You are given a 2D integer array

highways

where

highways[i] = [city1

i

, city2

i

, toll

i

]

indicates that there is a highway that connects

city1

i

and

city2

i

, allowing a car to go from

city1

i

to

city2

i

and vice versa

for a cost of

toll

i

You are also given an integer

discounts

which represents the number of discounts you have. You can use a discount to travel across the

i

th

highway for a cost of

toll

i

/ 2

(

integer

division

). Each discount may only be used

once

, and you can only use at most

one

discount per highway.

Return

the

minimum total cost

to go from city

0

to city

$n - 1$

, or

-1

if it is not possible to go from city

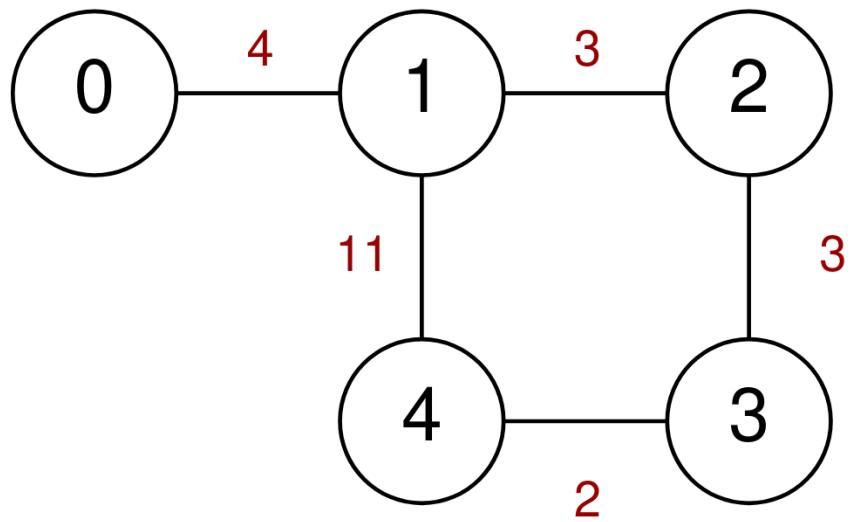
0

to city

$n - 1$

.

Example 1:



Input:

$n = 5$, highways = $[[0,1,4],[2,1,3],[1,4,11],[3,2,3],[3,4,2]]$, discounts = 1

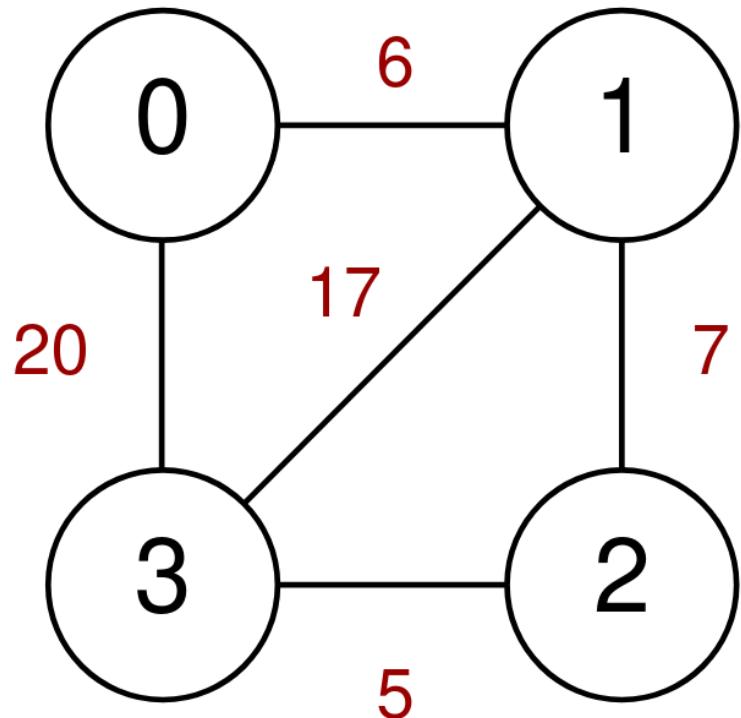
Output:

9

Explanation:

Go from 0 to 1 for a cost of 4. Go from 1 to 4 and use a discount for a cost of $11 / 2 = 5$. The minimum cost to go from 0 to 4 is $4 + 5 = 9$.

Example 2:



Input:

$n = 4$, highways = [[1,3,17],[1,2,7],[3,2,5],[0,1,6],[3,0,20]], discounts = 20

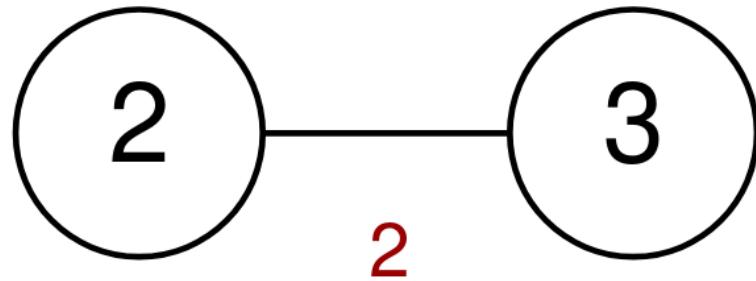
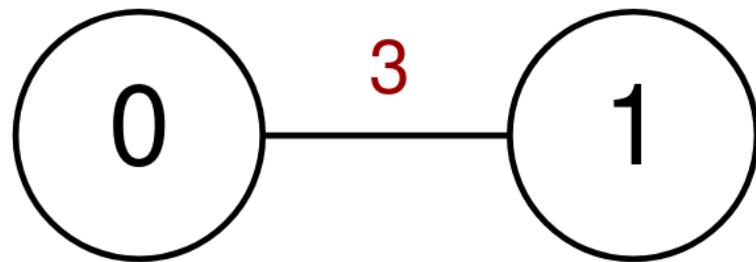
Output:

8

Explanation:

Go from 0 to 1 and use a discount for a cost of $6 / 2 = 3$. Go from 1 to 2 and use a discount for a cost of $7 / 2 = 3$. Go from 2 to 3 and use a discount for a cost of $5 / 2 = 2$. The minimum cost to go from 0 to 3 is $3 + 3 + 2 = 8$.

Example 3:



Input:

$n = 4$, highways = $[[0,1,3],[2,3,2]]$, discounts = 0

Output:

-1

Explanation:

It is impossible to go from 0 to 3 so return -1.

Constraints:

$2 \leq n \leq 1000$

$1 \leq \text{highways.length} \leq 1000$

$\text{highways}[i].length == 3$

$0 \leq city1$

i

, $city2$

i

$\leq n - 1$

$city1$

i

$\neq city2$

i

$0 \leq toll$

i

≤ 10

5

$0 \leq discounts \leq 500$

There are no duplicate highways.

Code Snippets

C++:

```
class Solution {
public:
    int minimumCost(int n, vector<vector<int>>& highways, int discounts) {
```

```
    }
};
```

Java:

```
class Solution {
public int minimumCost(int n, int[][][] highways, int discounts) {
    }
}
```

Python3:

```
class Solution:
def minimumCost(self, n: int, highways: List[List[int]], discounts: int) ->
    int:
```

Python:

```
class Solution(object):
def minimumCost(self, n, highways, discounts):
    """
    :type n: int
    :type highways: List[List[int]]
    :type discounts: int
    :rtype: int
    """
```

JavaScript:

```
/**
 * @param {number} n
 * @param {number[][][]} highways
 * @param {number} discounts
 * @return {number}
 */
var minimumCost = function(n, highways, discounts) {
};
```

TypeScript:

```
function minimumCost(n: number, highways: number[][], discounts: number):  
number {  
  
};
```

C#:

```
public class Solution {  
public int MinimumCost(int n, int[][] highways, int discounts) {  
  
}  
}
```

C:

```
int minimumCost(int n, int** highways, int highwaysSize, int*  
highwaysColSize, int discounts) {  
  
}
```

Go:

```
func minimumCost(n int, highways [][]int, discounts int) int {  
  
}
```

Kotlin:

```
class Solution {  
fun minimumCost(n: Int, highways: Array<IntArray>, discounts: Int): Int {  
  
}  
}
```

Swift:

```
class Solution {  
func minimumCost(_ n: Int, _ highways: [[Int]], _ discounts: Int) -> Int {  
  
}  
}
```

Rust:

```
impl Solution {  
    pub fn minimum_cost(n: i32, highways: Vec<Vec<i32>>, discounts: i32) -> i32 {  
        }  
    }  
}
```

Ruby:

```
# @param {Integer} n  
# @param {Integer[][][]} highways  
# @param {Integer} discounts  
# @return {Integer}  
def minimum_cost(n, highways, discounts)  
  
end
```

PHP:

```
class Solution {  
  
    /**  
     * @param Integer $n  
     * @param Integer[][][] $highways  
     * @param Integer $discounts  
     * @return Integer  
     */  
    function minimumCost($n, $highways, $discounts) {  
  
    }  
}
```

Dart:

```
class Solution {  
    int minimumCost(int n, List<List<int>> highways, int discounts) {  
        }  
    }
```

Scala:

```
object Solution {  
    def minimumCost(n: Int, highways: Array[Array[Int]], discounts: Int): Int = {
```

```
}
```

```
}
```

Elixir:

```
defmodule Solution do
  @spec minimum_cost(n :: integer, highways :: [[integer]], discounts :: integer) :: integer
  def minimum_cost(n, highways, discounts) do
    end
    end
```

Erlang:

```
-spec minimum_cost(N :: integer(), Highways :: [[integer()]], Discounts :: integer()) -> integer().
minimum_cost(N, Highways, Discounts) ->
.
```

Racket:

```
(define/contract (minimum-cost n highways discounts)
  (-> exact-integer? (listof (listof exact-integer?)) exact-integer?
    exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
 * Problem: Minimum Cost to Reach City With Discounts
 * Difficulty: Medium
 * Tags: array, graph, queue, heap
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
*/
```

```

*/



class Solution {
public:
    int minimumCost(int n, vector<vector<int>>& highways, int discounts) {
        }

    };
}

```

Java Solution:

```

/**
 * Problem: Minimum Cost to Reach City With Discounts
 * Difficulty: Medium
 * Tags: array, graph, queue, heap
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

class Solution {
    public int minimumCost(int n, int[][] highways, int discounts) {
        }

    }
}

```

Python3 Solution:

```

"""
Problem: Minimum Cost to Reach City With Discounts
Difficulty: Medium
Tags: array, graph, queue, heap

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
    def minimumCost(self, n: int, highways: List[List[int]], discounts: int) ->

```

```
int:  
# TODO: Implement optimized solution  
pass
```

Python Solution:

```
class Solution(object):  
    def minimumCost(self, n, highways, discounts):  
        """  
        :type n: int  
        :type highways: List[List[int]]  
        :type discounts: int  
        :rtype: int  
        """
```

JavaScript Solution:

```
/**  
 * Problem: Minimum Cost to Reach City With Discounts  
 * Difficulty: Medium  
 * Tags: array, graph, queue, heap  
 *  
 * Approach: Use two pointers or sliding window technique  
 * Time Complexity: O(n) or O(n log n)  
 * Space Complexity: O(1) to O(n) depending on approach  
 */  
  
/**  
 * @param {number} n  
 * @param {number[][]} highways  
 * @param {number} discounts  
 * @return {number}  
 */  
var minimumCost = function(n, highways, discounts) {  
};
```

TypeScript Solution:

```
/**  
 * Problem: Minimum Cost to Reach City With Discounts  
 * Difficulty: Medium
```

```

* Tags: array, graph, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function minimumCost(n: number, highways: number[][][], discounts: number): number {
}

}

```

C# Solution:

```

/*
* Problem: Minimum Cost to Reach City With Discounts
* Difficulty: Medium
* Tags: array, graph, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
    public int MinimumCost(int n, int[][][] highways, int discounts) {
        return 0;
    }
}

```

C Solution:

```

/*
* Problem: Minimum Cost to Reach City With Discounts
* Difficulty: Medium
* Tags: array, graph, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

```

```
int minimumCost(int n, int** highways, int highwaysSize, int*
highwaysColSize, int discounts) {
}
```

Go Solution:

```
// Problem: Minimum Cost to Reach City With Discounts
// Difficulty: Medium
// Tags: array, graph, queue, heap
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func minimumCost(n int, highways [][]int, discounts int) int {
}
```

Kotlin Solution:

```
class Solution {
fun minimumCost(n: Int, highways: Array<IntArray>, discounts: Int): Int {
}
```

Swift Solution:

```
class Solution {
func minimumCost(_ n: Int, _ highways: [[Int]], _ discounts: Int) -> Int {
}
```

Rust Solution:

```
// Problem: Minimum Cost to Reach City With Discounts
// Difficulty: Medium
// Tags: array, graph, queue, heap
```

```

// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
    pub fn minimum_cost(n: i32, highways: Vec<Vec<i32>>, discounts: i32) -> i32 {
        ...
    }
}

```

Ruby Solution:

```

# @param {Integer} n
# @param {Integer[][]} highways
# @param {Integer} discounts
# @return {Integer}
def minimum_cost(n, highways, discounts)

end

```

PHP Solution:

```

class Solution {

    /**
     * @param Integer $n
     * @param Integer[][] $highways
     * @param Integer $discounts
     * @return Integer
     */
    function minimumCost($n, $highways, $discounts) {

    }
}

```

Dart Solution:

```

class Solution {
    int minimumCost(int n, List<List<int>> highways, int discounts) {

```

```
}
```

```
}
```

Scala Solution:

```
object Solution {  
    def minimumCost(n: Int, highways: Array[Array[Int]]), discounts: Int): Int = {  
  
    }  
    }  
}
```

Elixir Solution:

```
defmodule Solution do  
  @spec minimum_cost(n :: integer, highways :: [[integer]], discounts ::  
  integer) :: integer  
  def minimum_cost(n, highways, discounts) do  
  
  end  
  end
```

Erlang Solution:

```
-spec minimum_cost(N :: integer(), Highways :: [[integer()]], Discounts ::  
integer()) -> integer().  
minimum_cost(N, Highways, Discounts) ->  
.
```

Racket Solution:

```
(define/contract (minimum-cost n highways discounts)  
(-> exact-integer? (listof (listof exact-integer?)) exact-integer?  
exact-integer?)  
)
```