11/27/2023:

<http://www.usaco.org/index.php?page=viewproblem2&cpid=1252>

K = [-----]

0123456789

.....X...X....X........X....X.X..............X...X..X...X........... <- cows

...........G.................G.....................G................ <- grass

[-----]

input:

K = distance

N = number of cows

5 9 ...

output:

M = minimum number of patches of grass

coordinates of the grass

import sys

**def optimize(my\_string, step\_size):**

**length = len(my\_string)**

# length = int(length)

# step\_size = int(step\_size)

result\_string = ["."] \* length

result = ""

found = False

if step\_size != 0 and length != 2:

for i in range(length):

found = False

L = max(i-step\_size, 0)

R = min(length, i+step\_size)

for j in *range(L, R):*

if result\_string[j] == my\_string[i]:

found = True

break

if found:

continue

for j in *range(R, L,* -1):

if result\_string[j] == ".":

result\_string[j] = my\_string[i]

break

else:

continue

count = 0

for i in result\_string:

if i != ".":

count += 1

result += i

return count, result

else:

return length, my\_string

def main():

test\_case = int(sys.stdin.readline())

length\_and\_step\_size = []

all\_my\_strings = []

for i in range(test\_case):

length\_and\_step\_size.append(list(map(int, sys.stdin.readline().strip("\n").split())))

all\_my\_strings.append(sys.stdin.readline().strip("\n"))

# [length, step\_size] = list(map(int, sys.stdin.readline().strip("\n").split()))

# my\_string = sys.stdin.readline().strip("\n")

# for i in optimize(my\_string, length, step\_size):

# sys.stdout.write(str(i) + "\n")

for l in range(test\_case):

[length, step\_size] = length\_and\_step\_size[l]

my\_string = all\_my\_strings[l]

**for w in optimize(my\_string, int(step\_size)):**

sys.stdout.write(str(w) + "\n")

main()

----------------------------------------------

.....X...**X**....X........X....X.X..............X...X..X...X........... <- cows

...........**G**........................................................ <- grass

...........G.................G.....................G................ <- grass

K = 5

cows = [5, 9, 14, 25, 30, 33, 50, 54, 57, 60]

answer = 1

current\_grass = cows[0] + K

for cow in cows:

if abs(cow - current\_grass) > K:

current\_grass = cow + K

answer += 1

print(answer)

K = 5

cows = [5, 9, 14, 25, 30, 33, 50, 54, 57, 60]

answer = 0

for cow in cows:

if (cow == cows[0]) **or** abs(cow - current\_grass) > K:

current\_grass = cow + K

answer += 1

print(answer)

------------------------------------------------

for i in range(N):

answer += i

print(answer)

for i in range(N):

answer += **i \* i**

print(answer)

for i in range(N):

answer += **i \* i**

print(answer)

--------------------------------------------------------------------------------------

import sys

def main():

T = int(sys.stdin.readline())

for \_ in range(T):

[N, K] = list(map(int, sys.stdin.readline().strip("\n").split()))

cows = sys.stdin.readline().strip("\n")

patches = 0

grass = ["."] \* N

prev\_h = -(K + 1)

prev\_g = -(K + 1)

for i in range(N):

cow = cows[i]

if cow == 'H':

if abs(i - prev\_h) > K:

**prev\_h = min(i + K, N - 1)**

if grass[prev\_h] != ".":

prev\_h -= 1

grass[prev\_h] = 'H'

patches += 1

else:

if abs(i - prev\_g) > K:

**prev\_g = min(i + K, N - 1)**

if grass[prev\_g] != ".":

prev\_g -= 1

grass[prev\_g] = 'G'

patches += 1

print(patches)

answer = ""

for g in grass:

answer += g

print(answer)

main()

======================================================================================

~50 2022, 2023 Bronze problems

======================================================================================

02/25/2024:

<https://judge.yosupo.jp/problem/static_range_sum>

L R

[------]

a = [\_, 3, 5, 2, 6, 3, 8, 5] # 1-indexed

pref\_sum = [0, 3, 8,10,16,19,27,32] # 0-indexed

def init():

pref\_sum = [0] \* (1 + len(a))

for i in range(1, len(a)):

pref\_sum[i] += pref\_sum[i - 1] + a[i]

return pref\_sum

def query(L, R):

return pref\_sum[R] - pref\_sum[L - 1]

--------------------------------------------------------------------------------------

<https://usaco.org/index.php?page=viewproblem2&cpid=572>

import sys

def main():

sys.stdin = open("bcount.in", "r")

sys.stdout = open("bcount.out", "w")

[N, Q] =list(map(int, input().strip("\n").split()))

cows\_list = []

for i in range(N):

cows\_list.append(int(input()))

temp = [0, 0, 0]

prefix\_arr = [[0,0,0]]

for i in range(N):

temp[cows\_list[i] - 1] += 1

prefix\_arr.append(temp.copy())

for k in range(Q):

[a,b] = list(map(int, input().strip("\n").split()))

result = []

for i in range(3):

result.append(prefix\_arr[b][i] - prefix\_arr[a - 1][i])

print(" ".join(list(map(str, result))))

main()

on-line vs off-line

on-line:

**6 3**

**2**

**1**

**1**

**3**

**2**

**1**

1 6 > 0 1 2

3 3 > 2 3 1

2 4 > 4 3 5

off-line:

**6 3**

**2**

**1**

**1**

**3**

**2**

**1**

1 6

3 3

2 4

0 1 2

2 3 1

4 3 5

--------------------------------------------------------------------------------------

<https://usaco.org/index.php?page=viewproblem2&cpid=595>

import sys

def main():

sys.stdin = open("div7.in", 'r')

sys.stdout = open("div7.out", 'w')

cows = []

for \_ in range(int(input())): # Time: O(N)

cows.append(int(input()))

pref\_sum = [0]

for cow in cows: # Time: O(N)

pref\_sum.append((pref\_sum[-1] + cow) % 7)

first\_occ = [0] \* 7

for i in range(len(pref\_sum) - 1, -1): # Time: O(N)

first\_occ[pref\_sum[i]] = i

last\_occ = [0] \* 7

for i in range(len(pref\_sum)): # Time: O(N)

last\_occ[pref\_sum[i]] = i

answer = -1

for i in range(7): # Time: O(1)

answer = max(answer, last\_occ[i] - first\_occ[i])

print(answer)

main()

--------------------------------------------------------------------------------------

<https://usaco.org/index.php?page=viewproblem2&cpid=762>

Assume all arrays are *1-indexed*.

ex.:

N = 5

a = [\_, 3, 1, **9, 2, 7**] # 1-indexed

^

k=3

suff\_min = starting from right to left, suffix minimum score

*pref\_sum = starting from left to right, prefix sum score*

def query(k):

return (*(pref\_sum[n] - pref\_sum[k - 1])* - suff\_min[k]) / (n - k)

<=>

suff\_min = starting from right to left, suffix minimum score

suff\_sum = starting from right to left, suffix sum score

import sys

def main():

sys.stdin = open("homework.in", 'r')

sys.stdout = open("homework.out", 'w')

n = int(input())

scores = list(map(int, input().split()))

scores.insert(0, 0)

suff\_sum = [0] \* (1 + n + 1)

suff\_min = [10000] \* (1 + n + 1)

for i in range(n, 0, **-1**):

suff\_sum[i] = suff\_sum[i + 1] + scores[i]

suff\_min[i] = min(suff\_min[i + 1], scores[i])

# suff\_max[i] = max(suff\_max[i + 1], scores[i])

# suff\_prod[i] = suff\_prod[i + 1] \* scores[i]

best\_score = [0, 1]

for k in range(n - 1, 1, **-1**):

#avg = (suff\_sum[k] - suff\_min[k]) / (n - k)

avg = [suff\_sum[k] - suff\_min[k], n - k]

# if best\_score < avg:

# if best\_score[0] / best\_score[1] < avg[0] / avg[1]:

if best\_score[0] \* avg[1] < avg[0] \* best\_score[1]:

best\_score = avg

answer = []

# elif best\_score == avg:

if best\_score[0] \* avg[1] == avg[0] \* best\_score[1]:

answer.append(k - 1)

answer.reverse()

print(**"\n"**.join(list(map(str, answer))))

main()

5

3 1 9 2 7

--------------------------------------------------------------------------------------

Assignment:

- <https://usaco.org/index.php?page=viewproblem2&cpid=691>

- <https://usaco.org/index.php?page=viewproblem2&cpid=715>

======================================================================================

02/25/2024:

<https://codeforces.com/gym/102951/problem/D>

import sys

from collections import deque

MAX\_VALUE = 10\*\*2

# Space complexity: O(MAX\_VALUE)

sums = [0] \* (1 + MAX\_VALUE)

n, q = list(map(int, input().split()))

# Time: O(N\*MAX\_VALUE)

for \_ in range(n):

L, R, addend = list(map(int, input().split()))

L += 1

R += 1

for i in range(L, R):

sums[i] += addend

pref\_sum = [0] \* (1 + MAX\_VALUE)

for i in range(1, MAX\_VALUE):

pref\_sum[i] = pref\_sum[i - 1] + sums[i]

for \_ in range(q):

L, R = list(map(int, input().split()))

L += 1

R += 1

print(pref\_sum[R - 1] - pref\_sum[L - 1])

------------------------------------------------

for \_ in range(n):

L, R, addend = list(map(int, input().split()))

L += 1

R += 1

for i in range(L, R):

sums[i] += addend

1 2 3 4 5 6 7 8 9

0 2 2 5 5 5 3 0 0

[------------] +2

[---------] +3

2 0 3 0 0 -2 -3 0 delta

0 2 2 5 5 5 3 0 0 ????

MAX\_VALUE = 10\*\*2

# Space complexity: O(MAX\_VALUE)

delta = [0] \* (1 + MAX\_VALUE)

sums = [0] \* (1 + MAX\_VALUE)

pref\_sum = [0] \* (1 + MAX\_VALUE)

# Time: O(N + MAX\_VALUE + Q)

for \_ in range(n):

L, R, addend = list(map(int, input().split()))

L += 1

R += 1

delta[L] += addend

delta[R + 1] += -addend

for i in range(1, MAX\_VALUE):

sums[i] = delta[i] + sum[i - 1]

pref\_sum[i] = pref\_sum[i - 1] + sums[i]

for \_ in range(q):

L, R = list(map(int, input().split()))

L += 1

R += 1

print(pref\_sum[R - 1] - pref\_sum[L - 1])

------------------------------------------------

1 2 3 4 5 6 7 8 9

0 2 2 5 5 5 3 0 0

[------------] +2

[---------] +3

1 3 2 4

changes = [[2, +2],

[7, -2],

[4, +3],

[8, -3]]

sorted\_changes =

[[2, +2],

[4, +3],

[7, -2],

[8, -3]]

import sys

n, q = list(map(int, input().split()))

changes = []

for \_ in range(n):

L, R, addend = list(map(int, input().split()))

L += 1

R += 1

changes.append([L, +addend, -1]) # delta[L] += addend

changes.append([R, -addend, -1]) # delta[R + 1] += -addend

for *query\_index* in range(q):

L, R = list(map(int, input().split()))

L += 1

R += 1

changes.append([L - 1, 0, query\_index, -1])

changes.append([R - 1, 0, query\_index, +1])

changes.sort(key = lambda x: [x[0], x[2]])

prev\_sum = 0

prev\_pref\_sum = 0

prev\_index = 0

**answers = [0] \* q**

for **change** in changes:

index = change[0]

sum = change[1] + prev\_sum

pref\_sum = sum + prev\_pref\_sum + prev\_sum \* (index - 1 - prev\_index)

prev\_sum = sum

prev\_pref\_sum = pref\_sum

prev\_index = index

if change[2] != -1:

query\_index = change[2]

answers[query\_index] += change[3] \* pref\_sum

for answer in answers:

print(answer)

----------------------------------------------

5 5

3 7 2

1 10 4

1 6 10

0 4 10

6 7 1

5 7

0 2

5 9

1 6

4 9

index: 0 1 2 3 4 5 6 7 8 9 10 11 12

delta: **+10** **+4** +2 -10 +1-10 -2 -4

**+10** -1

sum: 0 10 24 26 16 17 7

pref\_s: 0 10 34 84 110 143

126

[0, 0, 0]

[1, 10, 10]

[1, 10, 10]

[2, 14, 24]

[2, 24, 34]

[2, 24, 34]

[4, 26, 84]

[4, 26, 84]

[5, 26, **110**]

[5, 26, 110]

[6, 16, 126]

[6, 16, 126]

[7, 17, 143]

[7, 17, **143**]

[8, 7, 150]

[9, 5, 155]

[9, 4, 154]

[9, 4, 154]

[9, 4, 154]

[12, 0, 162]

sorted\_changes =

[[2, +2],

[4, +3],

[7, -2],

[8, -3]]

sum = 0 2 5 3 0

pref\_sum = 0 2 **7** 10 10

[------------] +2

[---------] +3

**0** 1 2 3 4 5 6 7 8 9 index

2 0 3 0 0 -2 -3 0 delta

**0** 0 **2** 2 **5** 5 5 **3** **0** 0 sum

**0** 0 **2** 4 **9** 14 19 **22** **22** 22 pref\_sum

9 = 5 + 2 + 2 \* (4-1-2)

22 = 3 + 9 + 5 \* (7-1-4)

pref\_sum = sum + prev\_pref\_sum + prev\_sum \* (index - 1 - prev\_index)

======================================================================================

03/03/2024:

<https://codeforces.com/problemset/problem/632/C>

xx**a**aaxxaaxxaxxx

xx**x**xxaxxaaxxaaa

[er, bcd, abba, abacaba]

[abacaba, abba, bcd, er]

c

cb

cba

cbacbc

a

ab

abc

aababc

-------------------------

dskj

dskjtd

dskj**d**skjtd

dskj**t**ddskj

dskj

dskjcd

dskj**c**ddskj

dskj**d**skjcd

A+B < B+A

class Word:

def \_\_init\_\_ (self, word):

self.word = word

def \_\_lt\_\_ (self, other):

return self**.word** + other.word < other.word + self.word

n = int(input())

strings = []

for \_ in range(n):

strings.append(Word(input().strip("\n")))

strings.sort()

for word in strings:

print(word.word, end="")

--------------------------------------------------------------------------------------

<https://usaco.org/index.php?page=viewproblem2&cpid=919>

*import.sys*

*sys.stdin = open("paintbarn.in", "r")*

*sys.stdout = open("paintbarn.out", "w")*

MAX\_XY = 1000

plane = []

for \_ in range(MAX\_XY):

plane.append([0] \* MAX\_XY)

(n, k) = list(map(int, input().split()))

for \_ in range(n):

(x1, y1, x2, y2) = list(map(int, input().split()))

for x in range(x1, x2):

for y in range(y1, y2):

plane[x][y] += 1

answer = 0

for x in range(MAX\_XY):

for y in range(MAX\_XY):

# print(plane[x][y], end=" ")

if plane[x][y] == k:

answer += 1

# print()

print(answer)

-------------------------------------------

delta:

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 **+1 0 0** **-1** 0 0 0 0

0 0 **0 0 0** 0 0 0 0 0

0 0 **0 0 0** 0 0 0 0 0

0 0 **-1** 0 0 **+1** 0 0 0 0

0 0 0 0 0 0 **0 0 0** 0

0 0 0 0 0 0 **0 0 0** 0

0 0 0 0 0 0 **0 0 0** 0

0 0 0 0 0 0 0 0 0 0

plane:

0 0 0 0 0 0 0 0 0 0

0 1 1 1 1 0 0 0 0 0

0 1 1 1 1 0 0 0 0 0

0 1 1 2 2 1 1 0 0 0

0 1 1 2 3 2 1 0 0 0

0 0 0 1 2 2 1 0 0 0

0 0 0 1 2 2 1 0 0 0

0 0 0 1 1 1 1 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

*import.sys*

*sys.stdin = open("paintbarn.in", "r")*

*sys.stdout = open("paintbarn.out", "w")*

MAX\_XY = 1000 + 1

delta = []

plane = []

for \_ in range(MAX\_XY):

delta.append([0] \* MAX\_XY)

plane.append([0] \* MAX\_XY)

(n, k) = list(map(int, input().split()))

for \_ in range(n):

(x1, y1, x2, y2) = list(map(int, input().split()))

delta[x1][y1] += +1

delta[x2][y1] += -1

delta[x1][y2] += -1

delta[x2][y2] += +1

plane[0][0] = delta[0][0]

for y in range(1, MAX\_XY):

plane[0][y] = delta[0][y] \

+ plane[0][y - 1]

for x in range(1, MAX\_XY):

plane[x][0] = delta[x][0] \

+ plane[x - 1][0]

for y in range(1, MAX\_XY):

plane[x][y] = delta[x][y] \

+ plane[x - 1][y] \

+ plane[x][y - 1] \

- plane[x - 1][y - 1]

answer = 0

for x in range(MAX\_XY):

for y in range(MAX\_XY):

# print(plane[x][y], end=" ")

if plane[x][y] == k:

answer += 1

# print()

print(answer)

plane:

***0 0 0 0 0* 0** 0 0 0 0

***0 0 0 0 0* 0** 0 0 0 0

***0 0 0 0 0* 0** 0 0 0 0

***0 0 0 0 0* 0** 0 0 0 0

***0 0 0 0 0* 0** 0 0 0 0

***0 0 0 0*** *t* **up** 0 0 0 0

***0 0 0 0 le* z** 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

z = delta[z] + up + *le* - **t**

--------------------------------------------------------------------------------------

<https://usaco.org/index.php?page=viewproblem2&cpid=1063>

class Cow:

def \_\_init\_\_(self, x, y):

self.x = x

self.y = y

n = int(input())

cows = []

for \_ in range(n):

(x, y) = list(map(int, input().split()))

cows.append(Cow(x, y))

# Coordinate compression x-wise

cows.sort(key = lambda cow : cow.x)

k = 0

for cow in cows:

cow.x = k

k += 1

# Coordinate compression y-wise

cows.sort(key = lambda cow : cow.y)

k = 0

for cow in cows:

cow.y = k

k += 1

answer = 0

for bottom in range(n):

for top in range(n):

# assert(cow[bottom].y == bottom)

# assert(cow[top].y == top)

xLeft = min(cow[bottom].x, cow[top].x)

xRight = max(cow[bottom].x, cow[top].x)

# assert(xLeft < xRight)

answer += queryCount(x=0..xLeft, y=bottom..top)

\* queryCount(x=xRight..n-1, y=bottom..top)

print(answer)

\*

333322221111000**\***------------------top

\*

\*

\*

\*

\*

\*

--------------------**\***00011112222--bottom

\*

\*

\*

for cow in cows:

cows[cow.x][cow.y] = 1

cowsSum[x][y] =def= how many cows are in (0,0)-(x,y)

def queryCount(x=0..xLeft, y=bottom..top):

return ?? cowsSum ??

# Hint: do some drawings!!!