K : Conveyor Belt

原案 : tokoharu
実装 : not
darsein
tokoharu

概要

- ベルトコンベアに沿ってN台のカウンターがあり、1..Nの名前がついている
- Q個のクエリがあり、A_iからB_i(A_i < B_i)へP_i個荷物をベルトコンベアで運んでほしいというクエリである
- ベルトコンベアは時刻0から稼働するが、初期状態は汚れていて使えない。
- 場所 i は i 秒後に使えるようになる。
- ベルトコンベアは時間 1につき1流れる。ベルトコンベアには単位長さにつき 1つ荷物を置ける
- それぞれの場所xの人は1秒に1つ荷物を置くことができるが、勝手に人の荷物を取れない。
- 自分宛ての荷物は当然とれる。同一カウンターでは取る業務と置く業務は同時にできない。
- これらの条件下で最適にベルトコンベアを活用するとき、全員に荷物が届くのは最小で何秒後か？
- この問題をクエリ 1...i の荷物がある設定でそれぞれの iに対して答えよ

- N, Q <= 10^5

例 (Sample Input 3)

N=5
1->3 : 3個
3->5 : 1個

e : 空
x : 使用不可

1クエリ目のこたえ

t=1 : 1xxxx

t=2 : 11xxx

t=3 : 111xx

t=4 : e11ex

t=5 : ee1ee

終了 : 答えは5

2クエリ目のこたえ

t=1 : 1xxxx

t=2 : e1xxx

t=3 : 1e1xx

t=4 : 112ex

t=5 : e112e

t=6 : ee1e2

終了 : 答えは6

解法(1)

- ベルトコンベアの1単位の一生を思い浮かべると、長さNの区間になる。
- 運ばれる荷物は区間ととらえることができるので、
  区間を被覆するような問題と捉えることが可能

- 問題は
- 1. 単純に被覆数を最小化するだけであれば終了時間と一致しないこと
- 2. そもそも被覆数の最小値はどうやって求めるのか

解法(2)

1. 単純に被覆数を最小化するだけであれば終了時間と一致しないこと

- これは新たに[2,N], [3,N], … といったダミーの荷物を加えることで実現可能
- 例 (N = 5 で3秒のとき)
  - 0秒目に地点0を出発 = [4,N] がダミー
  - 1秒目に地点0を出発 = [3,N] がダミー
  - 2秒目に地点0を出発 = [2,N] がダミー
- N秒まで、1秒ごとにダミーが増えていき、それ以後はダミーは必要ない
- 解がN秒くらいまではダミーとクエリを順次足していき、解を求める

解法(3)

2. そもそも被覆数の最小値はどうやって求めるのか

- 被覆したい対象は順序をつけることができる。
  - [a,b] < [c,d] ⇔ b < c
- 順序をつけれたときの被覆数の最小値と、
  反鎖の最大値は一致。(これは Dilworth’s Theorem)
- ここで言う反鎖は区間[x,x]を覆う区間[a,b]の個数となる。
  - 選んだすべての区間が [x,x]を覆うならこれらは反鎖
  - もし選んだすべての区間の共通部分が空ならば、
    選んだ区間の中に順序をつけれるものが存在する。

解法(4)

- まとめると、
  - 区間加算クエリと
  - 全体の最大値を求めるクエリ
- 両方を処理できるデータ構造があればよい。
- これはSegmentTreeがあれば実装できる。

ジャッジ解

darsein	C++	116 lines, 2.7kB
not	C++	55 lines, 1.4kB
tokoharu	C++	87 lines, 1.3kB

FA

hankan_rta 147:07

Sleep18e6 : 181:28