Meno: Tomáš Belan
Príklad 7**, kat.** O
Škola: ŠpMNDaG Teplická, Bratislava

O spracovaní vstupu. Mám pole riadky, kde mám všetky nájdené y súradnice, a tiež pole stlpce, kde mám všetky nájdené x súradnice. Tiež mám hash tabuľku bodky, ktorá uchováva pre každý vrchol nejaké údaje. Vybral som si hash tabuľku preto, že mi prišla vhodnejšia ako normálna matica: časová náročnosť sa blíži ku O(1), (aj keď najhorší prípad je O(N)), a pamäťová náročnosť je výrazne menšia, ako pri matici (kde každý bod, nie len vrchol, zaberá pamäť). Jazyk Ruby má štruktúru na hash tabuľky a vhodnú hash funkciu už vstavanú (zdrojové kódy sú na www.ruby-lang.org), čo robí moju prácu jednoduchšou. Ale keby z nejakého dôvodu nebolo možné hash použiť, stačí miesto všetkých „Hash.new" napísať „Array.new" a program bude stále fungovať, len to bude žrať strašne veľa pamäte.

Ako kľúče hash tabuľky používam celé čísla, kde dva bajty sú y a dva bajty sú x, v C-čkovom zápise x | (y<<16).

Najprv poriadne spracujem polia riadky a stlpce. Zoradím ich quicksortom[1] (ale je v podstate jedno, aký algoritmus použijem, pretože tam aj tak neskôr mám cyklus $O(N^2)$, takže časová náročnosť celého programu sa s algoritmom lepším ako $O(N^2)$ nezmení), a potom odstránim duplikáty (a keďže už sú zoradené, tak je to ľahké). Tým dostanem polia riadky_uniq a stlpce_uniq, v ktorých sú uložené tie x, resp. y súradnice, ktoré sú dôležité (tvar útvaru sa na nich nejako mení). Celý útvar sa rozdelí na niekoľko horizontálnych pásiem, kde každé ide od jednej takejto x súradnice po druhú. Podobne sú vertikálne pásma. Veľkosť týchto pásiem vypočítam a uložím do polí riad_pasmo a stlp_pasmo. Tieto pásma použijem hlavne pri rátaní obsahu.

A teraz ako sa zistí obvod n-uholníka: zoberme si ľubovoľnú priamku rovnobežnú so súradnicovou osou. Vrcholy na nej tvoria páry, a každý tento pár je spojený stranou. Zo zadania vyplýva, že od každého vrcholu ide jedna horizontálna a jedna vertikálna strana ku nejakým ďalším vrcholom. Tieto ďalšie vrcholy si pamätám v hash tabuľkách hor_spoje a ver_spoje. Mám cyklus, ktorý sa pre každú x súradnicu v stlpce_uniq, aké bodky na nej sú (na toto použije pole riadky_uniq), a každému páru nastaví navzájom údaje vo ver_spoje. (stlpce_uniq je dlhé najviac N, a podobne aj riadky_uniq, takže zložitosť cyklu je najviac $O(N^2)$). Potom nasleduje ďalší cyklus, ktorý podobne vyráta hor_spoje.

Potom vypíšem jednotlivé vrcholy: Vypíšem prvý vrchol. Potom sa presuniem na ten, s ktorým je horizontálne spojený, a vypíšem ho. Potom sa presuniem na ten, s ktorým je vertikálne spojený, a vypíšem ho, a tak ďalej.

Teraz ako vyrátať obsah: budem samostatne rátať obsah každého riadkového pásma. Každé riadkové pásmo má hodnotu sirka_riadku. To je číslo, ktoré určuje, aký by bol obsah toho pásma široký, keby sa všetky časti prilepili ku sebe. Celkový obsah riadkového pásma je jeho šírka krát jeho výška.

Teraz ako vyrátať šírku. Každý riadok má podobnú šírku, ako ten pred

1 Quicksort (popis samotného algoritmu viď minulé vzoráky KSP alebo wikipédia) má zložitosť väčšinou O(N log N), a v najhoršom $O(N^2)$. Ale to je v podstate jedno, pretože neskôr v programe je cyklus $O(N^2)$, ktorý určí zložitosť celého programu, a takýto „nevýznamný" sort, ak teda nie je ešte horší než $O(N^2)$, na nej nič nezmení.

ním, ale môže sa trochu zmeniť. Každá strana (čiže pár vrcholov) na danej priamke riadku šírku riadku zmení o svoju dĺžku. Ukážem to na druhom príklade v zadaní: prvé riadkové pásmo má šírku 3. Druhé riadkové pásmo má šírku podobnú ako 3, ale keďže je tam strana (0,1)-(1,1), štvorček (0,1)-(1,2) už nepatrí do vnútra n-uholníka, takže šírka druhého pásma sa o 1 zmenší. Podobne je tam strana (0,2)-(1,2), kvôli ktorej sa šírka druhého pásma tiež o 1 zmenší. Takže šírka druhého pásma bude 1.

Každá strana na danej riadkovej priamke môže ale šírku nasledujúceho riadku buď zväčšiť alebo zmenšiť. Keby sme ten príklad preklopili „hore nohami", tak má prvé pásmo šírku 1 a druhé pásmo sa o 2 rozšíri a bude mať šírku 3.

Pre každé stĺpcové pásmo si pamätám, či je v danom momente aktívne (či je súčasťou vnútra n-uholníka). Každá horizontálna strana n-uholníka buď má vnútro hore (a keďže obsah sa počíta smerom dole, ku zvyšujúcim sa y-súradniciam, tak svoje stĺpcové pásmo deaktivuje a šírku nasledujúceho riadkového pásma zmenší), alebo naopak, má vnútro dole (a šírku nasledujúceho riadkového pásma zväčší). (Poznámka: tu beriem, že smerom dole sa y zvyšuje.)

Strana môže byť samozrejme dlhá aj viacero stĺpcových pásiem, a šírka riadku sa zväčší/zmenší o šírku každého z nich.

Tu je moja implementácia popísaného algoritmu v Ruby.[2]

```
# toto ocakava napriklad takyto vstup:
# 0 1 2 2 3 0 0 0 3 1 2 1 1 2 1 1 (to je druhy priklad v zadani)
puts "Zadaj suradnice vsetkych bodov (oddelene medzerami)"
vstup = readline.split

raise "neni ich parny pocet" if vstup.length % 2 != 0
N = vstup.length / 2
riadky = Array.new; stlpce = Array.new
obsahy_riadkov = Array.new
bodky = Hash.new; hor_spoje = Hash.new; ver_spoje = Hash.new
N.times do |i|
  x = vstup[i*2].to_i; y = vstup[i*2+1].to_i
  stlpce << x; riadky << y   # pridam tieto suradnice do zoznamov
  # pouzivam integer kluc kde su dva bajty x a dva bajty y
  raise "suradnica sa nevojde do 2 bajtov" if y >= (1 << 16) or x >= (1 << 16)
  bodky[x | (y << 16)] = true
end

riadky.sort!; stlpce.sort!

riadky_uniq = Array.new; stlpce_uniq = Array.new
posledne = nil
riadky.each do |y|
  riadky_uniq << y if posledne != y
  posledne = y
end
stlpce.each do |x|
  stlpce_uniq << x if posledne != x
```

2 http://www.ruby.ch/interpreter/rubyinterpreter.shtml je online interpreter, s ktorým si môžete program vyskúšať. Nepodporuje vstup z konzoly, preto miesto 'vstup = readline.split' treba napísať priamo obsah poľa vstup, napr. 'vstup = [0,1, 2,2, 3,0, 0,0, 3,1, 2,1, 1,2, 1,1]'

```
  posledne = x
end

stlp_pasmo = Array.new; riad_pasmo = Array.new
(riadky_uniq.length - 1).times do |i|
  riad_pasmo << (riadky_uniq[i+1] - riadky_uniq[i])
end
(stlpce_uniq.length - 1).times do |i|
  stlp_pasmo << (stlpce_uniq[i+1] - stlpce_uniq[i])
end

# teraz sparujem kazdu bodku s tou, s ktorou je vertikalne spojena
stlpce_uniq.each do |x|
  spojena_y = nil
  riadky_uniq.length.times do |i|
    y = riadky_uniq[i]
    next if not bodky[x | (y << 16)]
    if spojena_y == nil
      spojena_y = y
    else
      y1 = spojena_y   # to, s cim tuto bodku sparujem
      ver_spoje[x | (y1 << 16)] = x | (y  << 16)
      ver_spoje[x | (y  << 16)] = x | (y1 << 16)
      spojena_y = nil
    end
  end
end

# podobne s ktorou je horizontalne spojena
# ale pritom vyratam aj obsah
# aktivne_pasmo ma hodnotu -1 pre neaktivne a 1 pre aktivne
# na zaciatku su vsetky pasma neaktivne
aktivne_pasmo = Array.new(stlpce_uniq.length, -1)
obsah = 0
sirka_riadku = 0
riadky_uniq.each_with_index do |y,riadok_i|
  spojena_x = nil; spojena_i = nil
  stlpce_uniq.length.times do |i|
    x = stlpce_uniq[i]
    next if not bodky[x | (y << 16)]
    if spojena_x == nil
      spojena_x = x; spojena_i = i
    else
      x1 = spojena_x   # to, s cim tuto bodku sparujem
      hor_spoje[x1 | (y << 16)] = x  | (y << 16)
      hor_spoje[x  | (y << 16)] = x1 | (y << 16)
      spojena_x = nil
      # a teraz vypocet obsahu
      # tato strana n-uholnika ide od pasma spojena_i az po pasmo i
      spojena_i.upto(i-1) do |j|
        aktivne_pasmo[j] = -aktivne_pasmo[j]
        # aktivne_pasmo[j] je -1 pri neaktivnom, takze ked je neaktivne,
        # tak sa sirka_riadku zmensi
        sirka_riadku += stlp_pasmo[j] * aktivne_pasmo[j]
      end
    end
  end
  obsah += riad_pasmo[riadok_i] * sirka_riadku
end

# vypisem body v poradi
# zacnem od bodu so suradnicami x=vstup[0] y=vstup[1]
teraz = vstup[0].to_i | (vstup[1].to_i << 16)
ide_hor_spoj = true
N.times do
```

```
  puts "(#{teraz & 0xFFFF},#{teraz >> 16})"
  # prejdem po horizontalnej ciare alebo po vertikalnej ciare
  teraz = (ide_hor_spoj ? hor_spoje[teraz] : ver_spoje[teraz])
  ide_hor_spoj = !ide_hor_spoj
end
puts "obsah je #{obsah}"
```