Laboratori Algorismia

G4 : 2.31

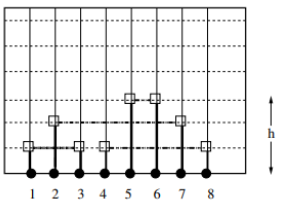
6 de octubre de 2022

1. Membres

Membres del grup:
- López Pazmiño, Ricardo
- Montalvo Falcón, Maria
- Walling Haro, Sven
- Pla Sanchis, Victor

2. Enunciat

2.31 Als circuits VLSI, s'utilitza un encaminament Manhattan sobre la placa aillant on va muntat el circuit. Les connexions horitzontals van per la cara de sota i les connexions verticals per la cara de sobre. Quan es necessiten connectar les connexions horitzontals amb les verticals, es perfora la placa amb el que s'anomena una via. Les connexions del circuit amb l'exterior es realitzen amb pins (veure la figura, on les connexions de sobre estan dibuixades en sòlid i les que van per sota amb línia discontínua). Tant les connexions horitzontals com verticals segueixen unes pistes dibuixades sobre la placa en forma d'una graella, i els pins estat alineats a un extrem de la placa. Sigui h el nombre de pistes horitzontals utilitzades. Si $L = (p1, q1), (p2, q2), ..., (pn, qn)$ són una seqüència de parells de pins a connectar (dos a dos), volem dissenyar un algorisme que connecti els parells de pins utilitzant el mínim nombre de pistes horitzontals h. Per exemple, considereu la següent figura:



Tenim com a entrada $L = (1, 3), (2, 7), (4, 8), (5, 6)$, el nombre de h és 3, i no es pot fer amb $h = 2$. En particular: dissenyeu un algorisme eficient, que donats n parells de pins, resolgui el problema de l'encaminament, de manera que es minimitzi h. Doneu-ne la complexitat i demostreu-ne la correctesa.

3. Solució

3.1. Algorisme

Quan ens referim al operador A[i:j] d'un conjunt A i de dos enters i i j, ens referim al subconjunt que va de $A[i]$ de manera inclusiva fins a $A[j]$ de manera exclusiva, com a exemple:

Operem: $A[0, 3]$
Entrada: $A = \{\ a, b, c, d, e, f\ \}$, $i = 0$, $j = 3$
Sortida: $\{\ a, b, c\ \}$
Cost: $O(1)$

Quan ens referim al metode $sortLeftPins(L)$, ens referim al metode que retorna el conjunt de pins L de manera que els primers elements de cada pin estan ordenats en ordre creixent, i que en cas de igualtat compara el segon element en ordre decreixent, com a exemple:

$sortLeftPins$
Entrada: $(4, 5), (1, 2), (3, 7), (3, 8)$
Sortida: $(1, 2), (3, 8), (3, 7), (4, 5)$
Cost: Com que l'algorisme l'hem imaginat com una petita modificació del merge sort, el seu cost en cas pitjor i en cas millor és $O(n \cdot log_2(n))$

```
function SOLVE(L)                    ▷ L = {(1,3),(2,7),(4,8),(5,6)} as an example.
    S ← sortLeftPins(L)              ▷ Cost of sorting O(n · log2(n))
    h ← 0                            ▷ h =Actual height, iterates from 0 to length of L
    while S not empty do             ▷ (a, b) constructs a pin.
        (a, b) ← S[0]
        S.remove((a, b))             ▷ We know that pin (a, b) can be connected on height h.
        for (u, v) in S do
            if b < u then
                (a, b) ← (u, v)
                S.remove((a, b))
            end if
        end for
        h ← h + 1
    end while
    return h                         ▷ Returns the min height solution.
end function
```

3.2. Cost i Correctessa

El millor cost de l'algorisme es produeix quan en una sola iteració del bucle *while* conectem tots els pins necessaris, per tant: $O(n)$ on tots els pins poden ser conectats a la mateixa altura (altura 0), el pitjor cas afegeix en cada iteració només un pin per altura i recorre L tants cops com pins hi ha, tot i això, en cada iteració recorre un pin menos (però assimptòticament no es veu reflexat): $O(n^2)$. S'ha d'afegir també el cost del metode $sortLeftPins$ cridat al inici de l'algorisme, per tant ens queda:

`Best case:` $O(n \cdot log_2(n)) + O(n) = O(n \cdot log_2(n))$
`Worst case:` $O(n \cdot log_2(n)) + O(n^2) = O(n^2)$

Pel que fa la correctessa de l'algorisme hem de demostrar que sempre donarem com a sortida l'altura minima necessària, dit d'altre manera, que a cada filera hi ha el màxim de connexións posibles. Cal veure que:

1. Donada una connexió en una filera qualsevol, hem d'assegurar que tant per la seva esquerra com per la seva dreta hi ha les màximes connexións posibles. En aquest cas, només ens caldrá comprovar-ho per la banda dreta, ja que tenim l'entrada ordenada de manera creixent.

2. Donada una connexió en una filera qualsevol, hem d'assegurar que si connectem una connexió a la seva dreta mai hi haurá una d'altre posible entre aquestes dues. Això ho podem afirmar pel mateix motiu que el punt 1, com que l'entrada està ordenada sempre agafarem la següent connexió més propera i posible.

D'aquesta manera cada filera tindrá el màxim de connexións posibles i l'altura final serà la mínima.