**DISSENY D’UN TECLAT**

# **Descripció de les estructures de dades i algorismes**

## **Projectes de Programació:** Quadrimestre Tardor 2023-24

## **Versió del lliurament: 1.1**

## **1ª Entrega**

**Grup 12.3:**

Pau Costafreda Tur

Xavier López Mañes

Hakima Marouan Marouan

Diego Rodriguez Milagro





# **Breu descripció de les estructures de dades i algorismes**

# HILL CLIMBING

## Explicació de les estructures de dades utilitzades

1. **teclatF**: Teclat que ens quedarà al final de l'execució de l'algoritme.
2. **teclat**: Teclat provisional que s'utilitzarà per millorar-lo i trobar un màxim local.
3. **teclatAux** i **teclatAux2**: Teclat al que se li ha aplicat un canvi des del teclat provisional.
4. **lletraFreq**: Mapa de les lletres de l'alfabet i el nombre de vegades que s'utilitzen.
5. **costLletres**: Matriu dels cops que va una lletra despres de l’altre en l’input de l’usuari.
6. **x** i **y**: Posició natural del polze en coordenades (x,y) dins del teclat.
7. **alfabet**: String que conté totes les lletres de l’idioma del teclat.
8. **porVisitar**: Cua de prioritat de teclats (resultants d'aplicar canvis a teclats ja visitats) els quals encara no s'han visitat. És a dir, que no s'han generat els teclats resultants d'aplicar canvis en aquests teclats. Aquests teclats estan organitzats de major a menor qualitat de solució, per així, quan haguem de generar més teclats, els creem a partir dels millors.
9. **visitados**: Conjunt de teclats que ja han estat afegits a porVisitar. Serveix per a no quedar-nos en un bucle a l'hora d'agafar el teclat amb millor qualitat de la cua de teclats visitats.
10. **heuristica**: Control de qualitat del teclat per minimitzar la distància entre les lletres que es troben més freqüentment juntes a les paraules.
11. **heuristica2**: Control de qualitat del teclat perquè les lletres més utilitzades es trobin més a prop d'on es troba el polze de l'usuari usualment.
12. **heuristicaTotal**: Control de calitat total. Es a dir, la suma de les dues heurístiques.

## Explicació de l’algorisme

Aquest algoritme ens serveix per reorganitzar un teclat amb l’objectiu de trobar una distribució que sigui més còmode i rapida d’utilitzar respecte a les paraules, frase o idioma introduït.

L'algorisme de hill climbing és cridat des de la classe de distribució en la qual s'haurà generat un teclat aleatori de l'idioma de l'input amb les dimensions sol·licitades. Es passa com a paràmetre (**teclat**) juntament amb el nombre de vegades que hi ha una lletra després d'una altra (**costLletres**), el nombre de vegades que apareix cada lletra (**lletraFreq**), la posició del polze dins del teclat (**x** i **y**) i l'alfabet seleccionat (**alfabet**).

A partir d'aquests paràmetres, es crea una cua de teclats per visitar (**porVisitar**) i un conjunt de teclats ja visitats (**visitados**). A aquestes dies estructures s'afegirà el teclat original. A partir d'aquí, es comença un bucle en el qual es buscarà en l'arbre de solucions per trobar la solució òptima.

En aquest bucle es parteix des de la solució actual, i des d'aquí es generen dos teclats més per cada lletra que es trobi al teclat. Un dels teclats es crea canviant la lletra actual per la que té a la seva dreta, i l'altre es genera canviant-la per la que té a baix. En cas que la lletra estigui en el borde dreta o borde de baix del teclat, es canviarà per la lletra respectiva de l'extrem contrari. Per tant, sigui n el número de lletres que hi ha a l’alfabet, el factor de ramificació serà O(2n).

Totes les solucions generades s'afegiran a **porVisitar,** per generar més endavant més solucions a partir de les noves, i a visitados, per si es tornen a generar aquestes solucions, no tornar-les a afegir a **porVisitar**. A més, per cada solució generada, es mantindrà a la variable **teclatF** si aquesta solució és la millor trobada fins el moment.

Per calcular quina de les solucions és millor, es farà amb la funció de **heurísticaTotal**, que és una combinació de **heurística** i **heurística2**. La primera heurística calcula com de prop estan les lletres que són adjacents en les paraules de l'input, i per tant quant més vegades vagin seguides dues lletres, més important serà que aquestes estiguin juntes al teclat. La segona heurística mesura com de lluny estan les tecles de la posició natural del polze. Novament, quan més vegades aparegui una lletra, la solució serà millor si aquesta està més a prop del polze.

L’algoritme va descobrint l’arbre de solucions, sempre obtenint els teclats “fills” del teclat amb millor heurística actual ja descobert. Com que podria estar infinitament descobrint solucions fins visitar totes les solucions possibles, se li afegeix un límit de 100 iterations. Al final d’aquestes iteracions es retornarà el teclat visitat (o “fill” de visitat) amb millor heurística.

Es posible que en una execució de hill Climbing, el millor teclat que es trobi no sigui molt bo, i és per això que desde la classe distribució es crida l’algoritme 10 cops amb 10 teclats aleatoris i es queda amb el millor.

En resum, l’algoritme rep un teclat amb una distribució aleatoria i desde aquí es van fent intercanvis en les posicions de les tecles fins que es troba la millor solució local. L’algoritme es crida varios cops per trobar una solució millor i entre les diferents solucions, es queda amb la millor.

## 