PROP 2017

Documentació de la pràctica

Membres del grup:

Adrià Pedraza Santos
Pol Linger
Nil Salvat
Bryan Ibarra

Per engegar el projecte:

1. A la fila 27 del fitxer Matriu_costos.java has de ficar la ruta del fitxer on estan definits els costos de les operacions inserir/borrar i substituir.

El format d'aquest fitxer serà de dues línies: la primera amb un número que indicarà el cost d'inserir i esborrar. La segona amb un número que indicarà el cost de substituir. Tal i com diu a l'enunciat de la pràctica.

2. A la fila 82 del fitxer Matriu_costos.java has de ficar la ruta del fitxer on estan els n genomes. No s'ha de modificar el segon paràmetre argv[i], només la ruta.

3. En la part esquerra del NetBeans, on t'apareixen tots els projectes, clic dret sobre el nostre projecte i clica a Properties (Propietats). A la llista fer clic sobre Run, i on fica Arguments has de ficar els n genomes, separats per espais en blanc. T'ho deixem per a que puguis fer copia/pega:

Bovi_n.txt Gallina_n.txt huma_n.txt Ovi_n.txt Ximpanze_n.txt Cocodril_n.txt Gat_n.txt macaco_n.txt porc_n.txt Conill_n.txt Gos_n.txt orangutan_n.txt Ratoli_n.txt

És important fer-ho en aquest ordre ja que sinó no et surtirà la mateixa matriu / arbre que el que t'hem enviat.

Un cop fet això ja pots executar el projecte.

Com hem programat la pràctica?

Hem utilitzat l'algorisme Neighbor-Joining (unió de veïns). Aquest algorisme el podriem classificar com algorisme Greedy.

És un algorisme iteratiu i que serveix justament per al que nosaltres voliem resoldre. En cada iteració s'uneixen dues espècies en una de nova, seguint unes pautes que ara explicarem.

Consta de 5 passos:

1. Calcular les distàncies promig de cada espècie en la matriu de distàncies. Per fer això tenim una fòrmula que bàsicament ens diu que hem de sumar totes les distàncies amb les altres espècies i dividir-ho entre el tamany de la matriu (en cada iteració disminueix en 1) – 2.

2. Construir una matriu intermèdia M que ens servirà per veure quines espècies hem d'ajuntar en cada iteració.
En la matriu M cada valor es calcula amb la fòrmula M[i][j] = matriu_distancies[i][j] - (ri + rj) on ri i rj són les distàncies promig de i / j calculades al pas 1.

3. Ens quedem amb el valor més petit de M i ajuntem les espècies indicades pel valor més petit en un nou node. Haurem de calcular les distàncies dels nodes antics (succesors) amb el nou node. Per calcular la distància i (distància del node esquerra) : matriu_distancies[i][j] + ri – rj . On ri i rj són les distàncies promig de i / j calculades al pas 1.

Per calcular la distància j (distància del node dreta) simplement farem matriu_distancies[i][j] - distanciaI. On distanciaI és la distància que acabem d'obtenir.

A més, haurem d'esborrar de la llista de genomes les espècies que acabem d'unir. Afegirem el nou node a la llista de genomes.

4. Fem càlculs de distàncies de tots els nodes antics amb el nou node acabat de crear.
Esborrem les files i columnes de les espècies ajuntades i afegim al final de la matriu de distàncies, una nova fila i columna del nou node, amb les distàncies calculades en aquest mateix pas.

Com abans, esborrarem de la llista de nodes/arbres els nodes ajuntats i afegirem el nou node.

Els 4 primers passos són iterats.

Quan arribem al pas 5 en la matriu de distàncies només tindrem dues espècies (que no són espècies com a tals, sinó una espècie que és un possible antecessor de totes les espècies que cobreix). Això es tradueix en que tenim dos subarbres i només haurem d'ajuntar-los

5. Creem l'arrel de l'arbre final i l'ajuntem amb els dos altres subarbres que tenim.

Ja tenim l'arbre evolutiu.

Diferències entre l'arbre evolutiu obtingut i la matriu

Amb lo explicat anteriorment aconseguim un arbre evolutiu bastant exacte respecte a la matriu de distàncies inicial.

A l'arbre, hi ha distàncies que són exactament igual que a la matriu. Hi ha alguns valors que no són exactes però si que s'aproximen molt.

Això és perquè a l'hora de fer els càlculs, al ser int (enters) hi ha divisions per exemple que no donen exacte i es perden decimals.

Uns quants exemples:

De Ximpanze a Macaco en la matriu de distàncies tenim 82 mentres que si fem la suma a l'arbre dóna 81.

De Cocodril a Conill en la matriu tenim 739 mentres que si fem la suma a l'arbre dóna 738.

De Cocodril a Orangutan en la matriu tenim 727 i en l'arbre 728.

De Macaco a Gat en la matriu tenim 262 i en l'arbre 263.

De Gat a Gos en la matriu tenim 207 i en l'arbre 207.

De Gallina a Ratolí en la matriu tenim 755 i en l'arbre 755.

De Ximpanze a Humà en la matriu tenim 12 i en l'arbre 12.

De Porc a Bovi en la matriu tenim 257 i en l'arbre 260.

De Gos a Conill en la matriu tenim 375 i en l'arbre 360.

De Ratolí a Ovi en la matriu tenim 447 i en l'arbre 449.

De Orangutan a Ximpanze en la matriu tenim 32 i en l'arbre 31.

De Porc a Cocodril en la matriu tenim 768 i en l'arbre 766.

De Bovi a Ovi en la matriu tenim 84 i en l'arbre 84.

De Gallina a Bovi en la matriu 756 i en l'arbre 766

Com veiem, els resultats són pràcticament exactes. La diferència més gran que hem trobat és de 3 (Porc-Bovi).

Els més exactes són els que estan més avall en l'arbre, és a dir, als últims nivells.

Quan més gran és la matriu més imprecisions hi han. Hem probat aquest mateix codi amb una matriu igual que la de l'enunciat (4x4) i allà si que són tots els números iguals.

Finalment ja tenim l'arbre, però no hem aconseguit una bona funció per imprimir-lo i per això t'hem enviat una foto de l'arbre fet a mà. Per dibuixar-ho hem seguit al peu de la lletra el que ens diu la funció.

La funció que tenim imprimeix l'arbre per nivells, des de l'arrel fins les fulles. El que s'ha de fer per aconseguir l'arbre és anar dibuixant els nodes amb les distàncies que ens diu. Els nodes a cada nivell estan ordenats d'esquerra a dreta.

Breu explicació de les classes utilitzades:

ArbreBinari.java

Per implementar l'arbre evolutiu farem servir la classe ArbreBinari.

Cada node tindrà: fill esquerra / fill dret / distància amb el fill esquerra / distància amb el fill dret / nom

Els nodes fulla tindràn distàncies 0 ja que no tenen fills.

El nom dels nodes intermedis hem decidit que sigui la concatenació dels dos fills. En teoria, no haurien de tenir nom però així és més fàcil d'entendre i de seguir l'arbre. També ens permet saber quines fulles cobreix cada node intermig. Per exemple, si tenim un node CocodrilGallinaRatoli podrem saber que aquest node cobreix les espècies Cocodril – Gallina – Ratolí.

El nom dels nodes fulla serà cadascuna de les espècies que tinguem a la matriu.

Especie.java

Classe molt simple que ens permetrà guardar la informació de cada espècie de la matriu. Cada espècie té un “adn” del tipus Genoma i un nom. Les dues operacions que tenim en aquesta classe (apart de la constructora) són per obtindre l’adn i el nom.

Genoma.java

La classe Genoma guardarà la informació de cada genoma. Cada genoma consta de dos Strings: una cadena de caràcters del tipus AGACTTGATACAATGA i un nom. Com abans, tenim solament dues operacions per obtenir la cadena i el nom.

ArbreEvolutiu.java

Implementa l’algorisme Neighbor-Joining descrit anteriorment i també algunes operacions auxiliars privades tals com esborrar/afegir una fila/columna a una matriu

Matriu_costos.java

Implementa l’algorisme LCS per obtenir la matriu de distàncies de n genomes.

Aquí és on està el main del projecte.

A part d’aquesta documentació també hem comentat bastant el codi per a que sigui fàcil d’entendre.