Búsqueda ascenso de colina

APLICACIÓN EN TEXTO:

INTRODUCCION

En la siguiente aplicación de realizará la búsqueda de una palabra en una frase o varias palabras.

Ejemplo:
Palabra a buscar: ***Rojo***
Frase: *Mi chompa es* ***roja***

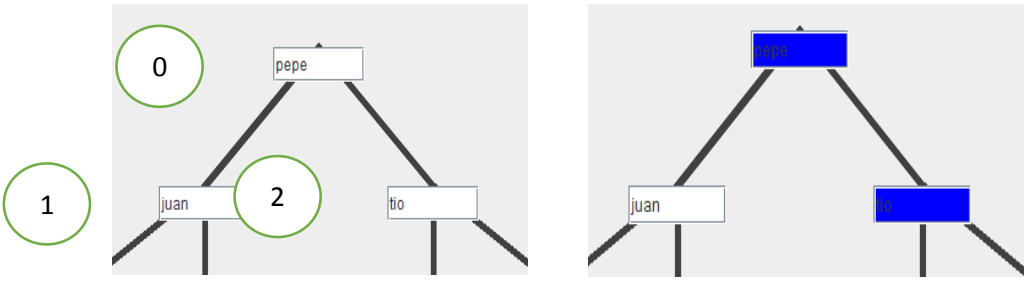
Utilizando la búsqueda ascenso en colina trataremos de encontrar la palabra Rojo en la frase, cada frase estará situada en un nodo de un árbol binario el cual nos complica más aun la búsqueda. Por esta razón utilizaremos reglas de búsqueda o heurísticas.
Las frases deben ingresarse en los nodos tal como lo aria una inserción por amplitud.

REGLAS

Regla 1.

En esta regla se seleccionara aleatoriamente si la cantidad de palabras en una frase son la misma que la otra y que no haya coincidencias de ninguna letra.

Ejemplo:



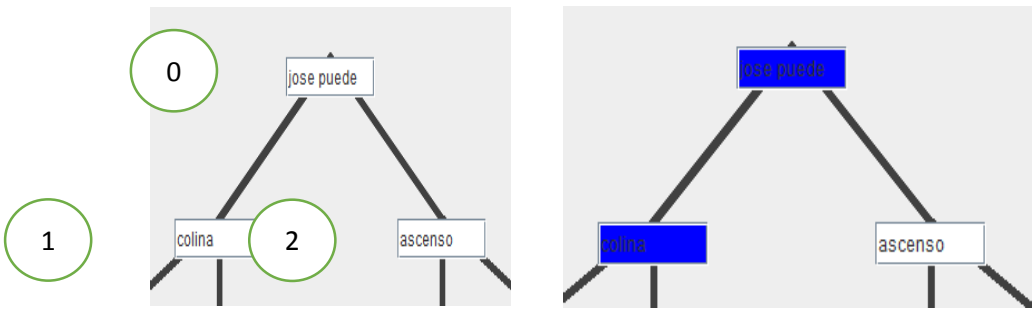
Meta: león

- Como no existe la palabra león en los nodos 1 y 2 así como la cantidad de palabras en las frases de cada nodo son iguales, tampoco hay coincidencias de las letras se selecciona aleatoriamente.
- Después de seleccionar el nodo 1 o 2 (en nuestro ejemplo seleccionó nodo 2) realiza la comparación con el nodo 0 para obtener el mejor nodo. En nuestro ejemplo volvió a seleccionar nodo 2 aplicando alguna regla. Nuestra ruta seria como lo ilustra la figura (derecha).

Regla 2.

En esta regla se selecciona por las cantidades de letras que coinciden con la palabra a buscar (meta).

Ejemplo:



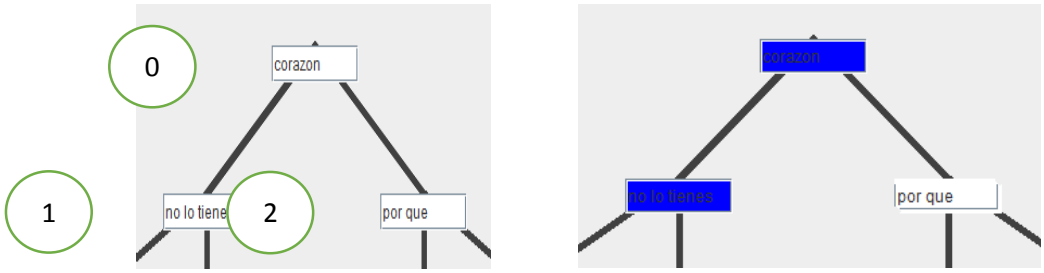
Meta: colegio

- Seleccionamos el nodo 1 porque coinciden en tres letras (col), ambos comienzan con col….. tanto como colegio y colina.
- Aplicando la misma regla el nodo 1 y nodo 0 vuelve a seleccionar nodo 1. Por esta razón nuestra ruta seria nodo 0-1 como lo ilustra la figura (derecha).

Regla 3.

En esta regla se selecciona la frase que tiene mayor cantidad de palabras en caso de que no haya coincidencias de letras.

Ejemplo:



Meta: juan

- Como en los nodos 1 y 2 no existe la palabra juan, entonces se escoge el nodo 1 porque tiene mayor cantidad de palabras (cantidad de palabras 3).
- Después de escoger el nodo 1 realiza la comparación con el nodo 0, de la cual obtiene el nodo mejor aplicando alguna regla.

IMPLEMENTACIÓN DE ALGORITMO

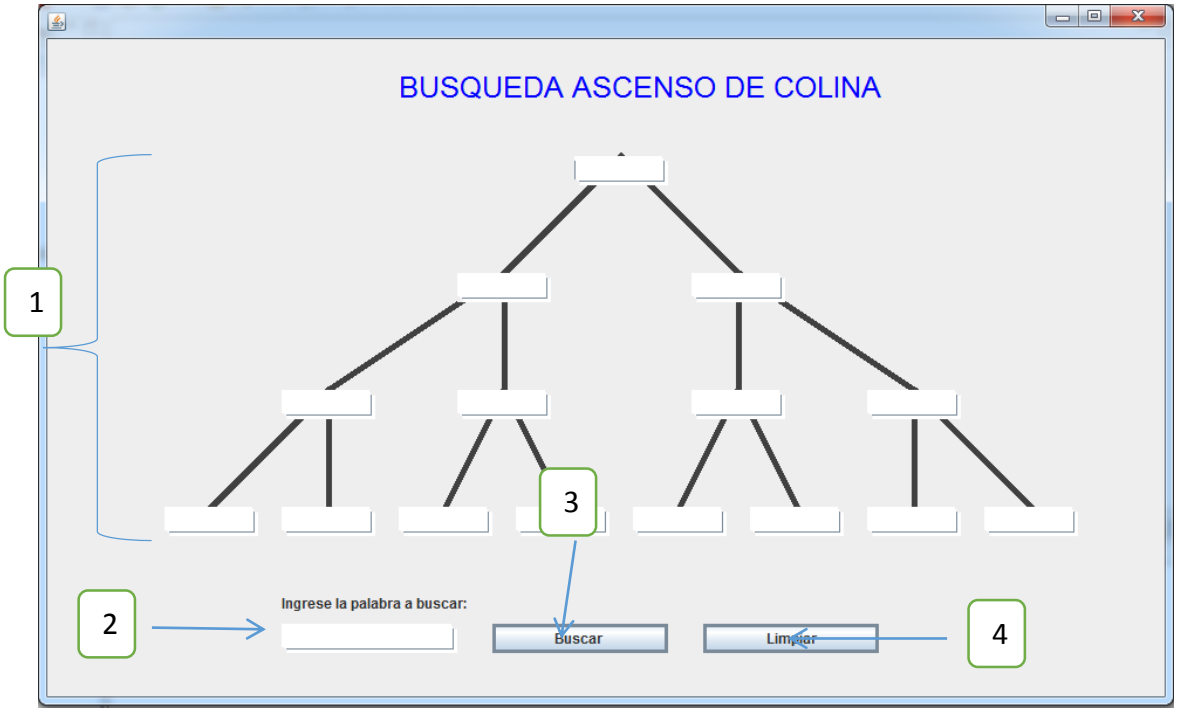
Algoritmo ascenso de colina

Actual= Estado_inicial
fin = falso
Mientras ¬fin hacer
Hijos= generar_sucesores(Actual)
Hijos= ordenar_y_eliminar_peores(Hijos, Actual)
si ¬vacio?(hijos) entonces
Actual=Escoger_mejor(Hijos)
si no fin=cierto
finMientras
finAlgoritmo

Algoritmo implementado en la aplicación

```
30  public void busqueda_ascenso_en_colina(String m){
31      meta = m;
32      boolean fin = false;
33      ruta = new Pila<Nodo>();
34      if(actualEsBuscado(arbol))
35          fin = true;
36      ruta.push(arbol.getRaiz());
37      Cola<ArbolB<Nodo>> actual_y_mejor_hijo = new Cola<ArbolB<Nodo>>();
38      ArbolB<Nodo> actual = arbol;
39      Cola<ArbolB<Nodo>> hijos = new Cola<ArbolB<Nodo>>();
40      Cola<ArbolB<Nodo>> copia_hijos = new Cola<ArbolB<Nodo>>();
41      ArbolB<Nodo> mejor_hijo = new ArbolB<Nodo>();
42      while(!fin){
43          hijos = generarSucesores(actual);
44          copia_hijos = generarSucesores(actual);
45          if(!vacio(copia_hijos)){
46              mejor_hijo = escoger_mejor_hijo(hijos);
47              actual_y_mejor_hijo.encolar(actual);
48              actual_y_mejor_hijo.encolar(mejor_hijo);
49              actual = escoger_mejor_hijo(actual_y_mejor_hijo);
50              if(!insertadoRuta(actual)){
51                  fin = true;
52              }
53          }
54          else
55                  fin= true;
56      }
57  }
```

INTERFAZ GRÁFICA DE APLICACIÓN



MANUAL USUARIO

1. Se deben ingresar las frases tal como lo aria una inserción por amplitud.
2. Ingrese la palabra que desea buscar en el árbol binario.
3. Presione el botón buscar para que muestre la ruta que encontró la búsqueda de ascenso en colina.
4. Si desea limpiar todo lo realizado presione el botón limpiar.

A continuación tenemos el algoritmo de hill-climbing o algoritmo de acenso a colina:

ALGORITMO HILL-CLIMBING

```
Algoritmo Hill Climbing

   Actual= Estado_inicial
   fin = falso

   Mientras ¬fin hacer
      Hijos= generar_sucesores(Actual)
      Hijos= ordenar_y_eliminar_peores(Hijos, Actual)
      si ¬vacio?(hijos) entonces
            Actual=Escoger_mejor(Hijos)
      si no fin=cierto
   finMientras

finAlgoritmo
```

Para una mejor comprensión de la búsqueda de acenso a colina tenemos el siguiente ejemplo implementado en JAVA.

2.- hacer una búsqueda implementando en el algoritmo de acenso a colina para ir desde cualquier provincia del departamento de Cochabamba asía otra provincia.

Mostrando en una red de rutas tenemos lo siguiente:

A continuación tenemos el algoritmo de implementado en java:

```
public void busquedaAscensoColina(){
    int cont=0;
    while(cont<3){
        System.out.println("raiz = "+raiz.getElemento());
        generarHijos(raiz,anterior);
        cont++;
    }
}
public void generarHijos(Nodo nr,String ant){
    String vec[]=vecinos.get(nr.getElemento());
    for (int k = 0; k < vec.length; k++) {
        if (vec[k].equals(ant)) {

        }
        else{
            System.out.println(""+vec[k]+"");
            raiz.addHijo(new Nodo(vec[k]));
        }
    }
    anterior=raiz.getElemento();
    raiz=getMejorHijo(raiz.getHijos());
}
```

```
public Nodo getMejorHijo(List<Nodo> hijos){
    Nodo mejor=hijos.get(0);
    for (int i = 1; i <hijos.size(); i++) {
        if (rutas.get(mejor.getElemento()+"-"+meta)>(rutas.get(hijos.get(i).getElemento()+"-"+meta))) {
            System.out.println(rutas.get(mejor.getElemento()+"-"+meta)+">"+rutas.get(hijos.get(i).getElemento()+"-"+meta));
            mejor=hijos.get(i);
        }
        else{
            System.out.println(rutas.get(mejor.getElemento()+"-"+meta)+">"+rutas.get(hijos.get(i).getElemento()+"-"+meta));
        }
    }
    return mejor;
}
```

El método acenso a colina lo que hace es imprimir el nodo raíz y luego expandir o generar sus hijos a partir desde la raíz imprimida anteriormente.

El generarHijos(Nodo nr,String ant) recibe un nodo y un string donde creamos un arreglo de strings vec[] y almacenamos los elementos vecinos del nodo.

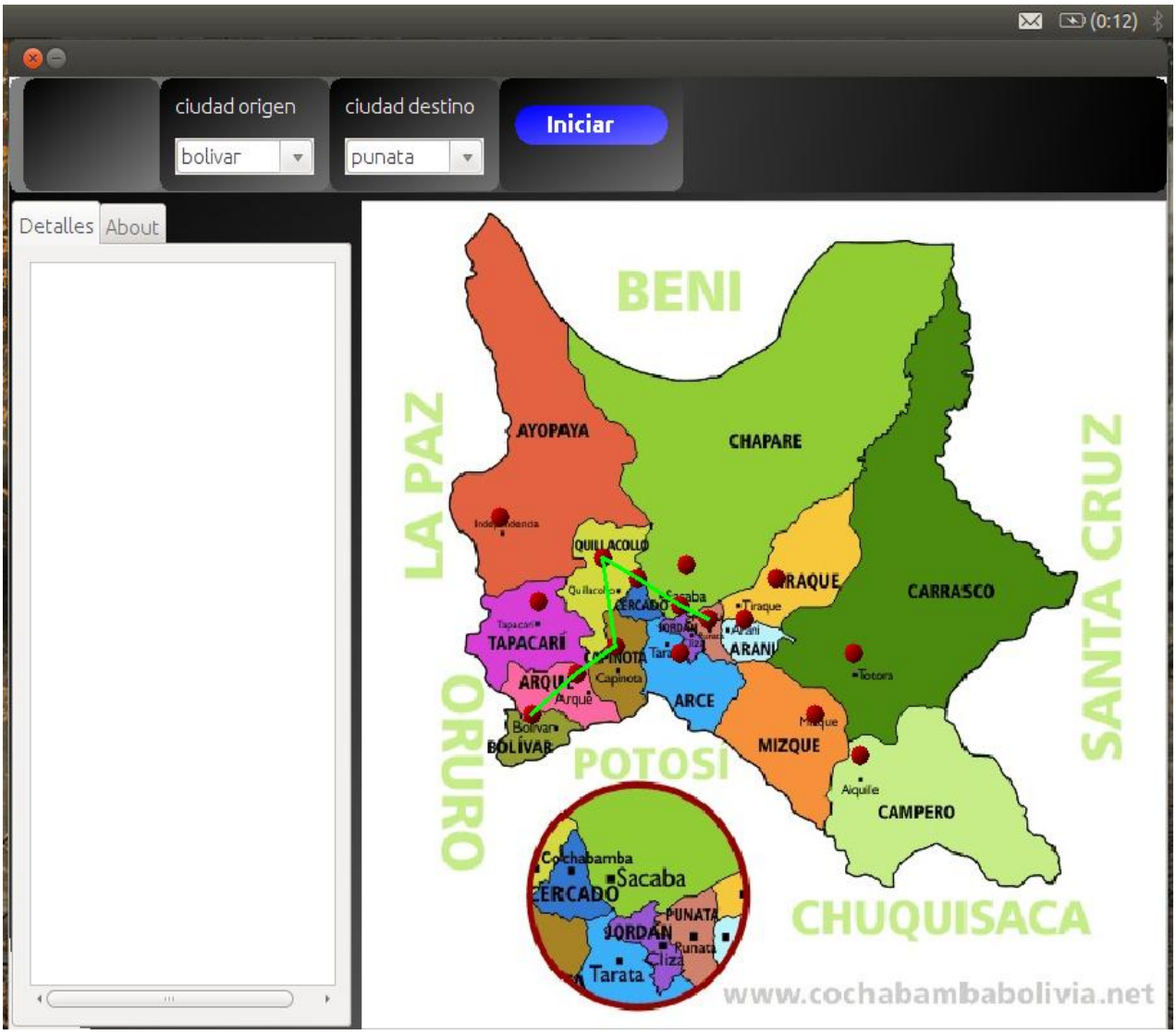
Recorremos con un for todo el arreglo de strings vec[] imprimiendo todos los hijos del de la raíz dada.

Después de imprimir todos los hijos llamamos el método getmejorhijo(parametro) y le enviamos como parámetro el nodo(vec[i]).

El método getMejorHijo(nodo(vec[i])) crea un mejorNodo donde el mejor nodo por defecto es el nodo de la posición(0) .

Recorremos con un for todos los elementos de la lista de nodos con la condición de que la nueva ruta sea mejor que la anterior y si es así la raíz será el nuevo mejor hijo obtenido y de esta nueva raíz expandimos sus nodos hijo.

Para una mejor comprensión de la aplicación tenemos la siguiente grafica del programa.



MANUAL DE USUARIO:

1.- primeramente ingresamos la provincia de origen en la que nos encontramos seleccionado de la lista desplegable.

2.- elegimos la provincia destino a la que queremos llegar en la lista desplegable de ciudad destino.

3.- luego damos iniciar y se inicia la búsqueda por ascenso de colina.

4.- en el cuadro de la izquierda en la pestaña de detalles nos muestra los caminos intermedios a recorrer en la búsqueda de ascenso a colina.

5.- y por último el programa nos grafica la ruta más corta a recorrer para llegar a nuestro destino.