Juego de la cuerda

Sumario

- Modelo de diseño Greedy
- Ejecución paso a paso
- Eficiencia

Modelo de diseño Greedy

Recordatorio:

Conjunto de Candidatos (C) : representa al conjunto de posibles decisiones que se pueden tomar en cada momento.

Conjunto de Seleccionados (S): representa al conjunto de decisiones tomadas hasta este momento.

Función Solución: determina si se ha alcanzado una solución (no necesariamente óptima).

Función de Factibilidad: determina si es posible completar el conjunto de candidatos seleccionados para alcanzar una solución al problema (no necesariamente óptima).

Función Selección: determina el candidato más prometedor del conjunto a seleccionar.

Función Objetivo: da el valor de la solución alcanzada.

Modelo de diseño Greedy

En nuestro caso:

Conjunto de candidatos (C): jugadores que aún no han sido seleccionados.

Conjunto de seleccionados (S): Los dos equipos formados por los jugadores ya seleccionados.

Función de solución: cuando no quedan jugadores sin equipo

Función de factibilidad: si hay más de 2 jugadores en el conjunto de candidatos a formar equipos.

Función de selección: selecciona al jugador con mayor fuerza de entre los jugadores no seleccionados.

Función objetivo: tener al final dos equipos lo más equilibrados posible.

Modelo de diseño Greedy

F. Factibilidad

Candidatos

Seleccionados (1&2)

F. Solucion

F. Seleccion

F. Objectivo

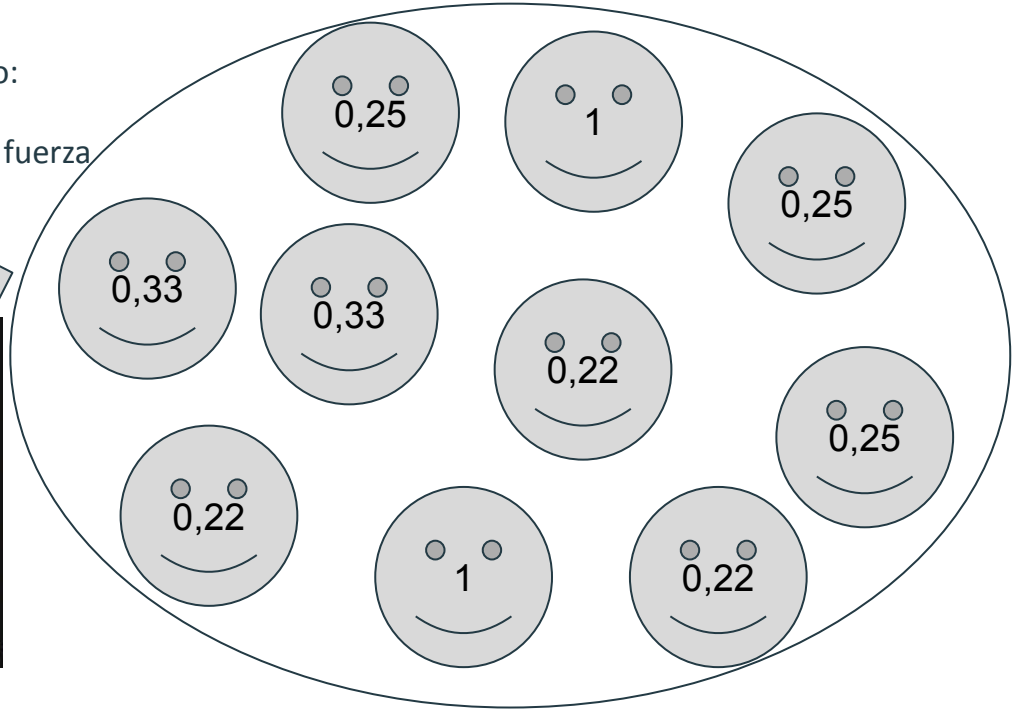
```
//Crear equipos equilibrados
void creaEquiposVoraz(vector<Concursante> concursantes, vector<Concursante>& equipo1, vector<Concursante>& equipo2){
    if(concursantes.size() < 2){
        cout << "No es possible de jugar con menos de 2 concursantes" << endl;
        return;
    }
    equipo1.push_back(concursantes[0]);
    equipo2.push_back(concursantes[1]);
    float fuerzaTotalEquipo1 = concur
    float fuerzaTotalEquipo2 = concu
    int i = 2;
    while(i < concursantes.size()){
        float fuerza = concursantes[i].getFuerza();
        if(fuerzaTotalEquipo2 <= fuerzaTotalEquipo1){
            equipo2.push_back(concursantes[i]);
            fuerzaTotalEquipo2 += concursantes[i].
        }
        else{
            equipo1.push_back(concursantes[i]);
            fuerzaTotalEquipo1 += concursantes[i].getFuerza();
        }
        i++;
    }
}
```

Ejecución paso a paso

Nuestro conjunto al inicio:

el número al centro es la fuerza

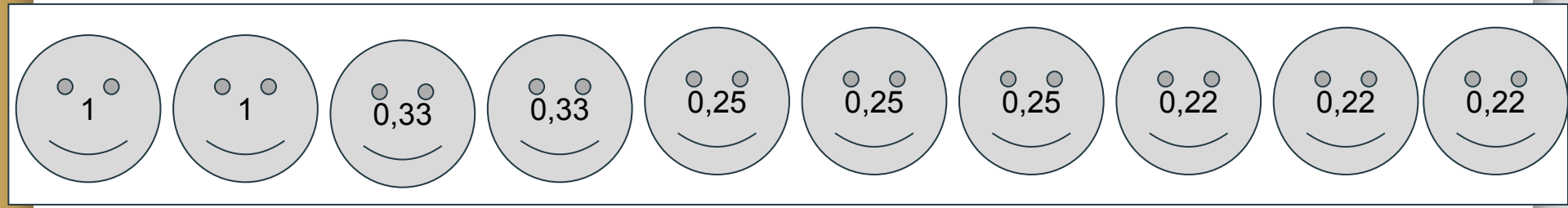
```
// rellenamos nuestra lista de competidores
concursantes.push_back(Concursante(1, 32, 70.5, true));
concursantes.push_back(Concursante(2, 45, 68.2, false));
concursantes.push_back(Concursante(3, 28, 80.3, true));
concursantes.push_back(Concursante(4, 50, 65.7, false));
concursantes.push_back(Concursante(5, 35, 72.1, true));
concursantes.push_back(Concursante(6, 42, 69.8, false));
concursantes.push_back(Concursante(7, 31, 73.6, true));
concursantes.push_back(Concursante(8, 47, 66.4, false));
concursantes.push_back(Concursante(9, 26, 78.9, true));
concursantes.push_back(Concursante(10, 39, 71.2, false));
```



Ejecución paso a paso

Clasificación de candidatos

```
 trieConcursante(concursantes, concursantes.size());
rt(concursantes.begin(), concursantes.end(), triDecroissant);
```



Ejecución paso a paso

```
// Ejecucion del algoritmo Greedy
creaEquiposVoraz(concursantes, equipo1, equipo2);
```

Ejecución paso a paso

Primera parte:

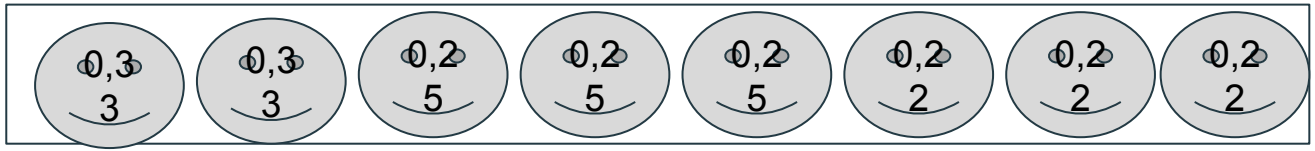
```
if(concursantes.size() < 2){
    cout << "No es possible de jugar con menos de 2 concursantes" << endl;
    return;
}
equipo1.push_back(concursantes[0]);
equipo2.push_back(concursantes[1]);
float fuerzaTotalEquipo1 = concursantes[0].getFuerza();
float fuerzaTotalEquipo2 = concursantes[1].getFuerza();
```

OK, size=10

Equipo 1: f1 = 0

Equipo 2: f2 = 0

no seleccionados



Ejecución paso a paso

Segunda parte:

```
int i = 2;
while(i < concursantes.size()){
    float fuerza = concursantes[i].getFuerza();
    if(fuerzaTotalEquipo2 <= fuerzaTotalEquipo1){
        equipo2.push_back(concursantes[i]);
        fuerzaTotalEquipo2 += concursantes[i].getFuerza();
    }
    else{
        equipo1.push_back(concursantes[i]);
        fuerzaTotalEquipo1 += concursantes[i].getFuerza();
    }
    i++;
```

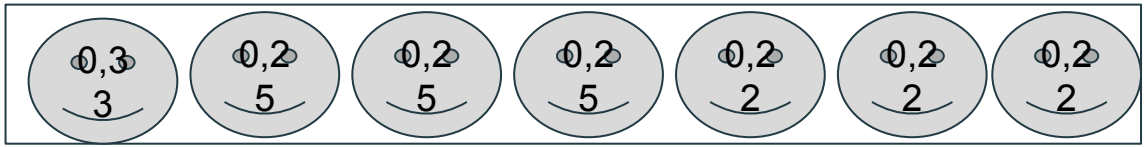
OK, f2 <= f1

i = 2

Equipo 1: f1 = 1

Equipo 2: f2 = 1

no seleccionados



Ejecución paso a paso

Segunda parte:

```
int i = 2;
while(i < concursantes.size()){
    float fuerza = concursantes[i].getFuerza();
    if(fuerzaTotalEquipo2 <= fuerzaTotalEquipo1){
        equipo2.push_back(concursantes[i]);
        fuerzaTotalEquipo2 += concursantes[i].getFue
    }
    else{
        equipo1.push_back(concursantes[i]);
        fuerzaTotalEquipo1 += concursantes[i].getFuerza();
    }
    i++;
```

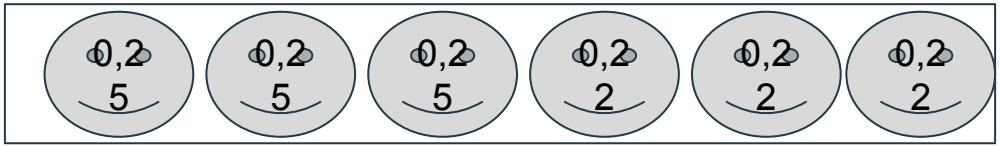
OK, f2 > f1

i = 3

Equipo 1: f1 = 1

Equipo 2: f2 = 1,33

no seleccionados



Ejecución paso a paso

Segunda parte:

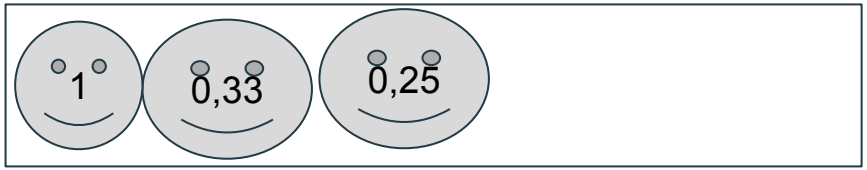
```
int i = 2;
while(i < concursantes.size()){
    float fuerza = concursantes[i].getFuerza();
    if(fuerzaTotalEquipo2 <= fuerzaTotalEquipo1){
        equipo2.push_back(concursantes[i]);
        fuerzaTotalEquipo2 += concursantes[i].getFuerza();
    }
    else{
        equipo1.push_back(concursantes[i]);
        fuerzaTotalEquipo1 += concursantes[i].getFuerza();
    }
    i++;
```

OK, f2 <= f1

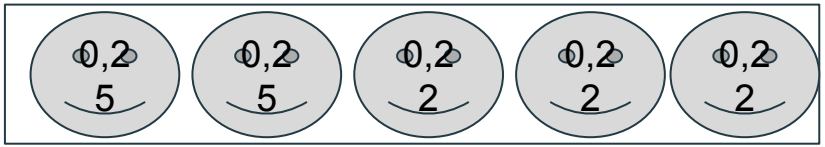
i = 4

Equipo 1: f1 = 1,33

Equipo 2: f2 = 1,33



no seleccionados



Ejecución paso a paso

Segunda parte:

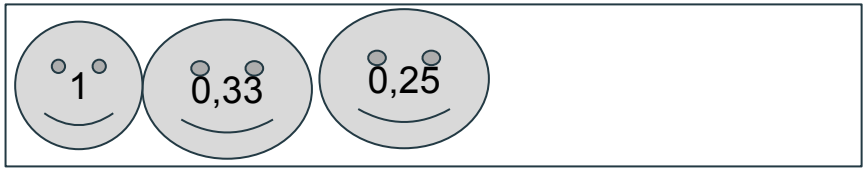
```
int i = 2;
while(i < concursantes.size()){
    float fuerza = concursantes[i].getFuerza();
    if(fuerzaTotalEquipo2 <= fuerzaTotalEquipo1){
        equipo2.push_back(concursantes[i]);
        fuerzaTotalEquipo2 += concursantes[i].getFue
    }
    else{
        equipo1.push_back(concursantes[i]);
        fuerzaTotalEquipo1 += concursantes[i].getFuerza();
    }
    i++;
```

OK, f2 > f1

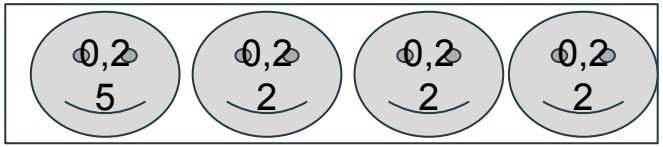
i = 5

Equipo 1: f1 = 1,33

Equipo 2: f2 = 1,58



no seleccionados



Ejecución paso a paso

Segunda parte:

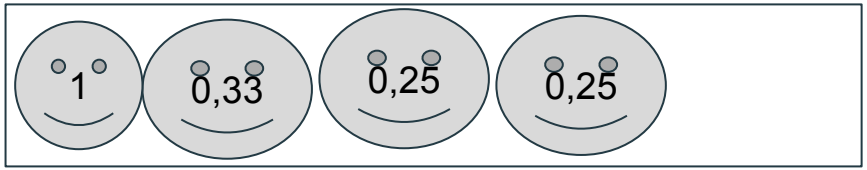
```
int i = 2;
while(i < concursantes.size()){
    float fuerza = concursantes[i].getFuerza();
    if(fuerzaTotalEquipo2 <= fuerzaTotalEquipo1){
        equipo2.push_back(concursantes[i]);
        fuerzaTotalEquipo2 += concursantes[i].getFuerza();
    }
    else{
        equipo1.push_back(concursantes[i]);
        fuerzaTotalEquipo1 += concursantes[i].getFuerza();
    }
    i++;
```

OK, f2 <= f1

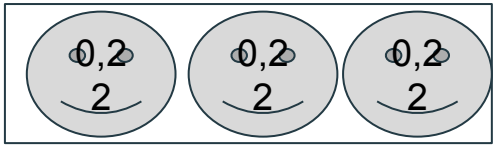
i = 6

Equipo 1: f1 = 1,58

Equipo 2: f2 = 1,58



no seleccionados



Ejecución paso a paso

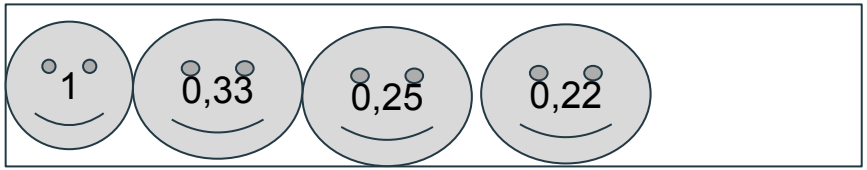
Segunda parte:

```
int i = 2;
while(i < concursantes.size()){
    float fuerza = concursantes[i].getFuerza();
    if(fuerzaTotalEquipo2 <= fuerzaTotalEquipo1){
        equipo2.push_back(concursantes[i]);
        fuerzaTotalEquipo2 += concursantes[i].getFue
    }
    else{
        equipo1.push_back(concursantes[i]);
        fuerzaTotalEquipo1 += concursantes[i].getFuerza();
    }
    i++;
```

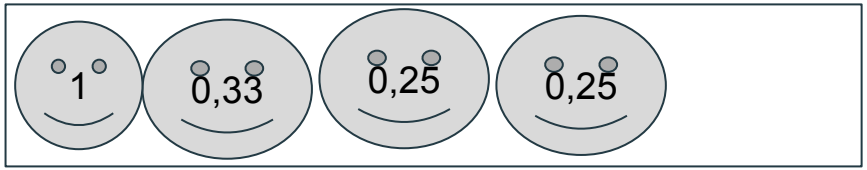
OK, f2 > f1

i = 7

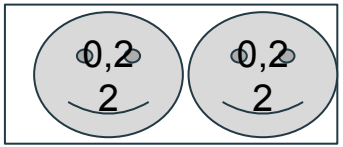
Equipo 1: f1 = 1,58



Equipo 2: f2 = 1,83



no seleccionados



Ejecución paso a paso

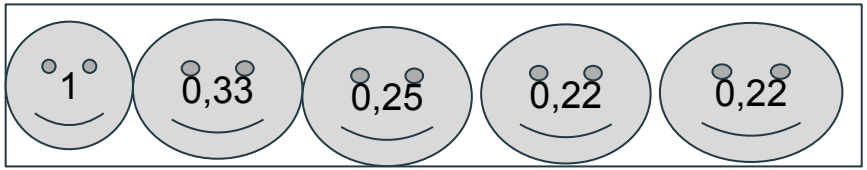
Segunda parte:

```
int i = 2;
while(i < concursantes.size()){
    float fuerza = concursantes[i].getFuerza();
    if(fuerzaTotalEquipo2 <= fuerzaTotalEquipo1){
        equipo2.push_back(concursantes[i]);
        fuerzaTotalEquipo2 += concursantes[i].getFue
    }
    else{
        equipo1.push_back(concursantes[i]);
        fuerzaTotalEquipo1 += concursantes[i].getFuerza();
    }
    i++;
```

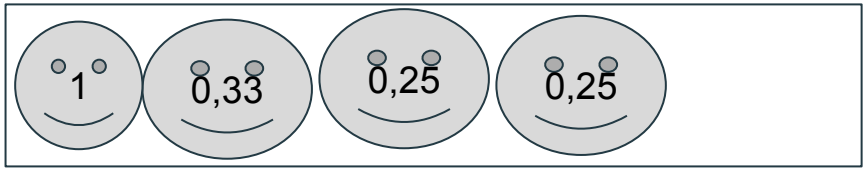
OK, f2 > f1

i = 8

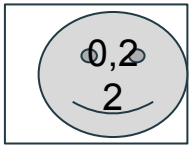
Equipo 1: f1 = 1,80



Equipo 2: f2 = 1,83



no seleccionados



Ejecución paso a paso

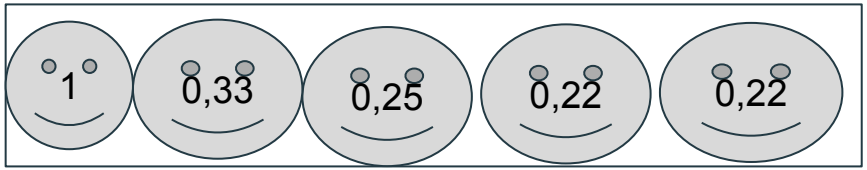
Segunda parte:

```
int i = 2;
while(i < concursantes.size()){
    float fuerza = concursantes[i].getFuerza();
    if(fuerzaTotalEquipo2 <= fuerzaTotalEquipo1){
        equipo2.push_back(concursantes[i]);
        fuerzaTotalEquipo2 += concursantes[i].getFuerza();
    }
    else{
        equipo1.push_back(concursantes[i]);
        fuerzaTotalEquipo1 += concursantes[i].getFuerza();
    }
    i++;
```

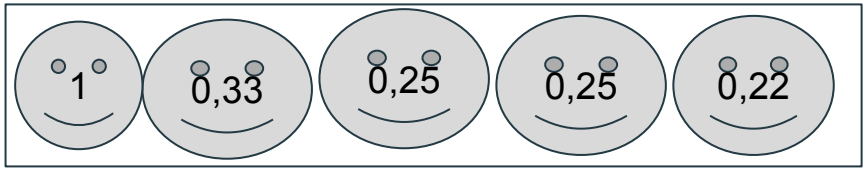
OK, f2 <= f1

i = 9

Equipo 1: f1 = 2,02



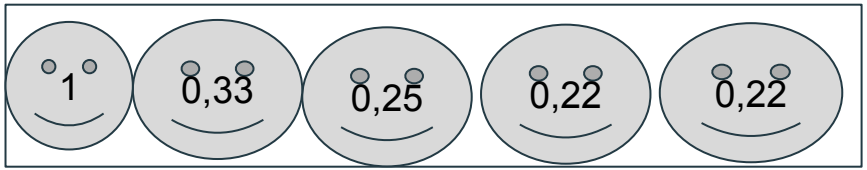
Equipo 2: f2 = 1,83



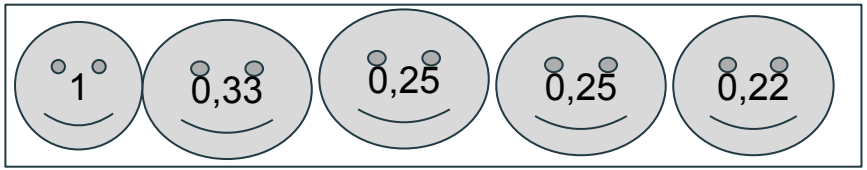
Ejecución paso a paso

Finalmente:

Equipo 1: f1 = 2,02



Equipo 2: f2 = 2,05




```
user@PNS-VirtualBox:~/Bureau/SI4/Algoritmica/ExerciceCM$ ./main
El tamano del equipo 1 es de 5
El tamano del equipo 2 es de 5

La fuerza total del equipo 1 es de: 2.00833
La fuerza total del equipo 2 es de: 2.04167
```

Eficiencia

1) Para la ordenación descendente de los concurrentes por fuerza, he utilizado la función sort de la biblioteca <algorithm>. La complejidad de esta función es O(n log n), donde n es el número de competidores de la lista.
2) Para la función creaEquiposVoraz, tenemos el bucle while que itera sobre todos los competidores desde la posición 2 hasta el final de la lista. La complejidad temporal de este bucle depende, por tanto, del tamaño de la lista de competidores, que es n. Las comparaciones y actualizaciones de valores en este bucle while son de complejidad constante. En consecuencia, la función creaEquiposVoraz es lineal, es decir, O(n).

   La eficiencia final del algoritmo es, por tanto, O(n log n).

Gracias