ParejasdeBaile

1. Tenemos dos conjuntos de personas y para cada persona sabemos su habilidad de baile. Queremos armar la máxima cantidad de parejas de baile, sabiendo que para cada pareja debemos elegir exactamente una persona de cada conjunto de modo que la diferencia de habilidad sea menor o igual a 1 (en módulo). Además, cada persona puede pertenecer a lo sumo a una pareja de baile. Por ejemplo, si tenemos un multiconjunto con habilidades $\{1, 2, 4, 6\}$ y otro con $\{1, 5, 5, 7, 9\}$, la máxima cantidad de parejas es 3. Si los multiconjuntos de habilidades son $\{1, 1, 1, 1, 1\}$ y $\{1, 2, 3\}$, la máxima cantidad es 2.

---

a) Considerando que ambos multiconjuntos de habilidades estan ordenados en forma creciente, observar que la solución se puede obtener recorriendo los multiconjuntos en orden para realizar los emparejamientos.

✓

---

b) Diseñar un algoritmo goloso basado en *a*) que recorra una única vez cada multiconjunto. Explicitar la complejidad temporal y espacial auxiliar.

```
public static int ParejasDeBaile(List<Integer> S, List<Integer> R) {
    int n = S.size();
    int res = 0;
    int j = 0;

    for (int k = 0; k < n && j < R.size(); k++) {
        if (Math.abs(S.get(k)-R.get(j)) <= 1) {
            j++;
            res++;
        }
    }
    return res;
}
```

Sean S, R los conjuntos ordenados de personas, $n = |S|$

El algoritmo se reduce a recorrer S, R hasta S_n o $j = |R|$, donde el i-ésimo paso decide si tenemos o no una pareja de baile.

La complejidad es trivialmente $O(|S| + |R|)$ temporal, y $O(1)$ espacial.

---

c) Demostrar que el algoritmo dado en *b*) es correcto.

Solución:

Tenemos los multiconjuntos ordenados de personas

$$S = \{s_1, ..., s_n\}$$

$$R = \{r_1, ..., r_m\}$$

Nuestra solución greedy G, y una solución óptima O, tal que nuestra solución toma una pareja como válida si y solo si la diferencia del módulo es menor o igual a 1, o sea:

$$(s_i, r_j) \iff |s_i - r_j| \leq 1$$

Vemos que G no puede ser mejor que O (absurdo), nos alcanza con probar que G no es peor que O, de tal forma que sea igual de óptima que O.

$$G = \{(s_1, r_1)...(s_h, r_h)\}$$

$$O = \{(s'_1, r'_1)...(s'_h, r'_h)\}$$

Por inducción fuerte

Hipótesis inductiva: Asumimos que existe solución óptima O', con $O'_j = G_j, \forall k \geq j \geq 1$

Caso base

Si $k = 0$, $(|O'| = |O| = 0) \Rightarrow O' = O$

Paso recursivo

Queremos probar que $k \to k + 1$, o sea, $G_{k+1} = (s_{k+1}, r_{k+1})$

Hasta ahora tenemos que $\{(s_i, r_i), ..., (s_i, r_i)\}$ son los primeros i elementos de O' (o sea, los mismos que en G)

Y queremos ver qué pasa en $k + 1$

Vemos O:

Si $(s_{k+1}, r_{k+1}) \in O$ ya constituye una solución óptima.

Caso contrario, tenemos 2 chances:

1. (s_{k+1}, r') con $r' > r_{k+1}$
2. (s', r_{k+1}) con $s' > s_{k+1}$

Queremos ver que pasa si intercambiamos (1) o (2) por (s_{k+1}, r_{k+1}) en O'

Caso 1

(s_{k+1}, r') por (s_{k+1}, r_{k+1}) :

Por G sabemos que (s_{k+1}, r_{k+1}) es pareja válida, luego r' queda sin elegir, y como $r' > r_{k+1}$ podrá ser elegida por un $s_p, p > k + 1$ en pasos siguientes, por lo que el número de parejas válidas se mantiene igual. $|O'_{k+1}| = |G_{k+1}|$

Caso 2

(s', r_{k+1}) por (s_{k+1}, r_{k+1}) :

G se empareja s_{k+1}, r_{k+1} si vale $|s_{k+1}, r_{k+1}| \leq 1$

Y tenemos que si $(s', r_{k+1}) \in O'$ entonces $s' > s_{k+1}$, y S está ordenado, notamos que entonces $|s' - r_{k+1}| > |s_{k+1} - r_{k+1}|$ y la única chance de que G no haya emparejado s_{k+1}, r_{k+1} es que $|s_{k+1} - r_{k+1}| > 1$, por lo que no es válido.

Por lo tanto: $\forall O :: \exists O'$ óptima tal que $G_k \subseteq O' \implies G$ es óptima.