Algoritmos golosos (Greedy)

March 30, 2018

Contenidos

Qué es un algoritmo goloso?

- Los algoritmos golosos resuelven problemas de optimización (ej: minimizar un cierto costo, o maximizar una cierta ganancia).
- Un algoritmo se dice goloso si toma una decisión en cada paso (basada en cierto criterio), reduciendo el problema a una instancia más chica.

Ejemplo - problema de la moneda

Enunciado

Dado un monto X en centavos, decir la menor cantidad de monedas (de valores 100, 50, 25, 10, 5, 1) son necesarias para pagarlo exactamente (asumiendo que tengo cantidad infinita de cada tipo).

Ejemplo - problema de la moneda

Enunciado

Dado un monto X en centavos, decir la menor cantidad de monedas (de valores 100, 50, 25, 10, 5, 1) son necesarias para pagarlo exactamente (asumiendo que tengo cantidad infinita de cada tipo).

Solución

Tomar siempre la moneda de valor más grande que no se pasa de X.

Algunas consideraciones

- Las decisiones que toman los algoritmos golosos se asumen óptimas, y no se revisan más adelante.
- No todos los problemas admiten una solución greedy. Sin embargo, es muy común creer que una idea greedy (errónea) funciona, pero finalmente el problema se soluciona con una técnica más sofisticada.
- Lo ideal es demostrar que la idea greedy es correcta antes de implementarla.
- Sin embargo, muchas veces las ideas greedy son difíciles de demostrar, por lo que por lo general es suficiente con tener una intuición de "por qué anda".

Problema de la moneda (continuación)

- Volviendo al problema de la moneda, veamos qué pasa si los valores de las monedas no están dados en el enunciado, sino que son input.

Ejemplo

Monedas: 1, 3, 4. Valor a pagar: 6.
El greedy respondería [4,1,1], pero la solución óptima es [3,3] (el greedy ya no funciona).

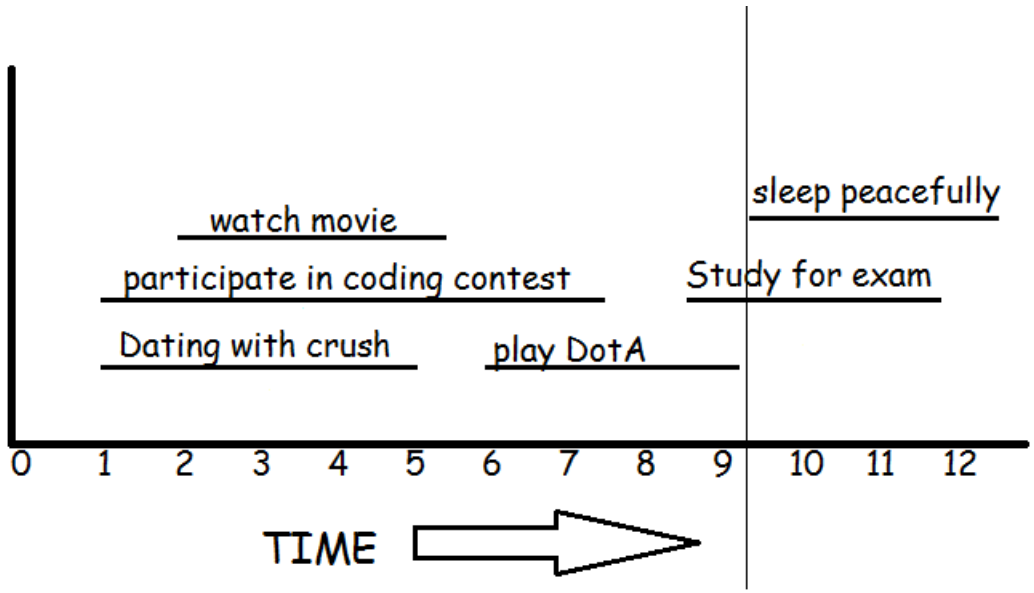
- Como veremos más adelante, este problema se resuelve mediante programación dinámica.

Máxima cantidad de intervalos que no se intersecan

www.spoj.com/problems/BUSYMAN/

Enunciado

Tengo planeadas n actividades. Cada una tiene un tiempo de inicio a_i y un tiempo de finalización b_i. Decir la mayor cantidad de actividades que puedo realizar (es decir, no se deben intersecar sus respectivos intervalos de tiempo, pero puedo empezar una actividad inmediatamente después de terminar otra).



Máxima cantidad de intervalos que no se intersecan

- Solución: Tomar siempre el intervalo que termina antes (entre los que puedo tomar).
- Por qué anda: Supongamos que tengo una solución óptima que no toma el que termina antes (llamémoslo k). Es fácil convencerse de que si reemplazo el primer intervalo de esta solución por el k, entonces formo otra solución óptima, la cual contiene el k. Luego, existe una solución óptima que toma la decisión de agregar el k.

Solución - código

```
int solve(int a[], int b[], int n){
  vector<pair<int,int>> intervals;
  for(int i = 0; i < n; ++i)
    intervals.push_back({b[i],a[i]});
  sort(intervals.begin(), intervals.end());
  int res = 0, last_end = -1;
  for(int i = 0; i < n; ++i){
    int start = intervals[i].second;
    int end = intervals[i].first;
    if(start >= last_end){
      res++;
      last_end = end;
    }
  }
  return res;
}
```

Ganar, gustar y galantear

www.spoj.com/problems/TAP2014G/

Enunciado

Germán y Gianina organizaron un partido de fútbol y quieren formar los equipos eligiendo una vez cada uno quién juega en su equipo. Hay N (par) jugadores y cada uno tiene un valor de habilidad a_i. Germán elige primero, y quiere que su equipo tenga una suma de habilidades mayor que la del equipo de Gianina. Para quedar bien, en algunos turnos le deja elegir primero a Gianina. ¿Cuál es la mayor cantidad de turnos en la que puede dejar que elija Gianina de modo tal de asegurarse tener un equipo mejor que el de ella?

- Ejemplos y solución: En pizarrón.

Reemplazar dígitos minimizando suma

`http://codeforces.com/contest/910/problem/C`

Enunciado

Dada una expresión de la forma $s_1 + s_2 + \cdots + s_n$, donde los s_i son strings de caracteres entre 'a' y 'j', reemplazar cada letra por un dígito decimal distinto, de modo que los s_i no empiecen con 0, y se minimice el resultado de evaluar la expresión.
Por ejemplo, si la expresión es " ab + de + aj ", una solución óptima es $10 + 23 + 14 = 47$

- Solución: En pizarrón

Conclusión

- No hay mucha teoría o algoritmos generales detrás de greedy, sino que cada problema tiene su particularidad. Por eso el énfasis en los ejemplos.
- No hay otra forma de aprender greedy, más que resolviendo problemas que salgan con greedy :D