PENSANDO EN GREEDY

Programacion Competitiva

Friday 22º February, 2019

Santiago Hincapie Potes

Universidad EAFIT

EL DIA DE HOY VEREMOS

1. Algoritmos greedy

2. Greedy is god

3. Proxima sesion

ALGORITMOS GREEDY

¿QUE SON?

→ Diremos que un algoritmo es greedy cuando en cada paso, elige la “mejor” solución local.

¿QUE SON?

- → Diremos que un algoritmo es greedy cuando en cada paso, elige la “mejor” solución local.
- → Dicha función de elección puede conducirnos o no a una solución óptima.

¿QUE SON?

- → Diremos que un algoritmo es greedy cuando en cada paso, elige la “mejor” solución local.
- → Dicha función de elección puede conducirnos o no a una solución óptima.
- → Cuando el algoritmo conduzca a una solución óptima diremos que el greedy “funciona”.

¿QUE SON?

- → Diremos que un algoritmo es greedy cuando en cada paso, elige la “mejor” solución local.
- → Dicha función de elección puede conducirnos o no a una solución óptima.
- → Cuando el algoritmo conduzca a una solución óptima diremos que el greedy “funciona”.
- → Beneficio inmediato.

PROBLEMA DE LA MONEDA

→ El problema de cambio de monedas aborda la forma de encontrar el número mínimo de monedas (de ciertas denominaciones) tales que entre ellas suman una cierta cantidad.

PROBLEMA DE LA MONEDA

- El problema de cambio de monedas aborda la forma de encontrar el número mínimo de monedas (de ciertas denominaciones) tales que entre ellas suman una cierta cantidad.
- Elegimos en cada paso la moneda de mayor denominacion que no supere el monto.

PROBLEMA DE LA MONEDA

- El problema de cambio de monedas aborda la forma de encontrar el número mínimo de monedas (de ciertas denominaciones) tales que entre ellas suman una cierta cantidad.
- Elegimos en cada paso la moneda de mayor denominacion que no supere el monto.
- ¿Funciona esta idea?

PROBLEMA DE LA MONEDA

- El problema de cambio de monedas aborda la forma de encontrar el número mínimo de monedas (de ciertas denominaciones) tales que entre ellas suman una cierta cantidad.
- Elegimos en cada paso la moneda de mayor denominacion que no supere el monto.
- ¿Funciona esta idea?
- Consideremos que tenemos monedas de $(25, 15, 1)$ y deseamos dar un cambio de 30

PROBLEMA DE LA MONEDA

- El problema de cambio de monedas aborda la forma de encontrar el número mínimo de monedas (de ciertas denominaciones) tales que entre ellas suman una cierta cantidad.
- Elegimos en cada paso la moneda de mayor denominacion que no supere el monto.
- ¿Funciona esta idea?
- Consideremos que tenemos monedas de $(25, 15, 1)$ y deseamos dar un cambio de 30
- El algoritmo encontraria la secuencia $\{25, 1, 1, 1, 1, 1\}$, sin embargo, la secuencia optima es $\{15, 15\}$

PROBLEMA DE LA SELECCIÓN DE TAREAS

Juan tiene n actividades que realizar y sabe cuándo empieza y cuándo termina cada una. Lamentablemente algunas se superponen y por lo tanto no puede realizarlas todas. El problema pide la máxima cantidad de actividades que Juan puede realizar sin que se le superpongan dos de ellas.

- → Por ejemplo si tenemos tres tareas de rangos $(1, 3)$, $(2, 9)$ y $(8, 10)$
- → . . . la respuesta sería 2 tareas, la primera y la última.

PROBLEMA DE LA SELECCIÓN DE TAREAS

→ ¿Hay alguna forma de decidir rápidamente qué tarea hacer primero?

PROBLEMA DE LA SELECCIÓN DE TAREAS

- → ¿Hay alguna forma de decidir rápidamente qué tarea hacer primero?
- → ¿elegir la tarea que dure menos tiempo?

PROBLEMA DE LA SELECCIÓN DE TAREAS

- → ¿Hay alguna forma de decidir rápidamente qué tarea hacer primero?
- → ¿elegir la tarea que dure menos tiempo?
- → ¿la tarea que empiece primero?

PROBLEMA DE LA SELECCIÓN DE TAREAS

- → ¿Hay alguna forma de decidir rápidamente qué tarea hacer primero?
- → ¿elegir la tarea que dure menos tiempo?
- → ¿la tarea que empiece primero?
- → No funcionan.

PROBLEMA DE LA SELECCIÓN DE TAREAS

- → ¿Hay alguna forma de decidir rápidamente qué tarea hacer primero?
- → ¿elegir la tarea que dure menos tiempo?
- → ¿la tarea que empiece primero?
- → No funcionan.
- → **Clave:** Escoger la tare que te deje el mayor tiempo posible para realizar las próximas.

PROBLEMA DE LA SELECCIÓN DE TAREAS

- → La forma correcta de ordenarlas es por horario de finalización
- → Siempre que podamos realizar la próxima tarea la realizamos, sino la ignoramos.
- → De esta forma, intuitivamente vamos realizando una a una las tareas con el objetivo de que nos sobre mayor tiempo para realizar las otras.
- → ¿Funciona esto?

¿POR QUÉ ES CORRECTO ESTE ALGORITMO?

- → Supongamos que el algoritmo no es óptimo
- → Con la selección de tareas que nosotros realizamos vamos resolviendo los siguientes subproblemas: ¿Cuántas actividades podemos hacer desde que terminaron las primeras i actividades? ¿Cuál es la próxima tarea a hacer?
- → Supongamos que en ese subproblema, no hay solución eligiendo como primer tarea la que finaliza primero dentro de las posibles.
- → Borremos la primer tarea elegida, y pongamos la que finaliza primero de las posibles. Todas las otras claramente van a poder realizarse
- → Por lo tanto hay una solución óptima que elije la primer tarea que finaliza. Contradicción.
- → Luego, el algoritmo es óptimo

TEOREMA DE NICO ALVAREZ

"Todos los problemas Greedies salen igual. Hay que ordenar 'las tareas' y después resolverlas en ese orden. Para ver en que orden se resuelven tenes que agarrar dos tareas y ver cual es la que greedymente se tiene que hacer primero"

TEOREMA DE NICO ALVAREZ

"Todos los problemas Greedies salen igual. Hay que ordenar 'las tareas' y después resolverlas en ese orden. Para ver en que orden se resuelven tenes que agarrar dos tareas y ver cual es la que greedymente se tiene que hacer primero"Eso quiere decir que el código será simplemente:

- → Hacer una función de comparación entre 2 tareas
- → Ordenar el 'arreglo de tareas'
- → Hacer un `for`

La parte más difícil claramente es la función de comparacion

GREEDY IS GOD

PROXIMA SESION

CONTEST

TODO

NEXT WEEK

¿Como son los problemas de una maraton de programacion?