Trabajo Práctico 2: Programación Dinámica para el Reino de la Tierra

Facultad de Ingeniería de la Universidad de Buenos Aires

Teoría de Algoritmos

Cátedra Buchwald-Genender

Gómez Belis, Sofía
Padrón: 109358
email: sgomezb@fi.uba.ar

Llanos Pontaut, Valentina
Padrón: 104413
email: vllanos@fi.uba.ar

Orsi, Tomas Fabrizio
Padrón: 109735
email: torsi@fi.uba.ar

Indice

1 Análisis del problema

1.1 Descripción y objetivo

TO-DO

1.2 Análisis y ecuación de recurrencia

La resolución de un problema por medio de la programación dinámica implica reutilizar las soluciones a subproblemas más pequeños en problemas más grandes que los incluyan. En el contexto actual, el foco está puesto en maximizar la cantidad de enemigos eliminados dados `n` minutos. Esta variable `n` es crucial en el análisis del problema planteado puesto que si se tiene `k` minutos, con $k < n$, necesariamente la solución será menor o igual (solo son iguales si no llegan enemigos en el minuto `n` y $n = k - 1$) a la solución en el minuto `n`. De esta forma, la cantidad de minutos que tiene el ataque repercute en el resultado final del combate. Por ejemplo, si $n = 0$, se puede afirmar que la cantidad de enemigos eliminados será también 0. Entonces, nuestros subproblemas estarán dados por la cantidad máxima de enemigos que se pueden eliminar en cada minuto $i \leq n$.

Sabiendo la forma de los subproblema, debemos analizar cómo se componen para resolver subproblemas más grandes. Si queremos obtener la solución óptima en el minuto `i`, vamos a poder utilizar las soluciones parciales calculadas hasta entonces, pero no nos interesa si existen o no problemas más grandes. Si estamos en el minuto $i = n$, sabiendo que es el último, tenemos dos estrategias posibles: atacar o cargar. ¿Tiene sentido cargar sabiendo que no se puede atacar a más enemigos? No. Esto se debe a que la cantidad de enemigos que se podrá eliminar en ese minuto será mayor o igual a 0, pero si cargamos energía, será definitivamente nula. Por lo tanto, en el último minuto conviene siempre atacar. Ahora bien, como establecimos antes, en el minuto $i \leq n$ no importa si $i = n$ o $i < n$, solamente debemos calcular el óptimo actual. En base al análisis previamente presentado, siempre va a ser mejor atacar a cargar, más allá de que en la solución final (problema mayor) se lleve a cabo la estrategia opuesta debido a que eso esté contemplado en el óptimo del minuto n.

Si en el minuto `i` los Dai Li atacan, la cantidad de enemigos eliminados en ese instante será $min(f(j), x_i)$. El valor de x_i es conocido, pero la energía acumulada por la policía secreta de la ciudad depende de los minutos que pasaron desde el último ataque. De esta manera, tenemos una segunda variable involucrada en el problema: `j`. Si el último ataque fue realizado hace un minuto, actualmente se podrá eliminar $min(f(1), x_i)$ enemigos y la cantidad máxima acumulada será la suma entre este valor y el correspondiente valor en el ataque anterior. ¿Qué sucede si el óptimo hace dos minutos es mayor que en el minuto anterior o su suma con $min(f(2), x_i)$ lo es? Como queremos mazimizar el resultado final, claramente nos conviene haber atacado hace dos minutos, lo cual también indica que en el minuto $i-1$ se cargó energía. Tenemos varias opciones para el ataque anterior, más precisamente $1 \leq j \leq i$, pero utilizaremos aquel

que nos lleve a la mejor solución. Entonces, el óptimo para el minuto `i` será la suma entre $min(f(j), x_i)$ y el óptimo en el minuto $i - j$.

Sabiendo la forma de los subproblemas y la manera en que éstos se combinan, podemos plantear la ecuación de recurrencia para el minuto `i`:

$opt[i] = max(min(f(j), x_i) + opt[i - j])\forall j \in [1; i]$

Como caso base, tenemos que en el minuto 0 se eliminan 0 enemigos.

Encontrada la ecuación de recurrencia, prodecemos a aplicarla iterativamente de manera bottom up, construyendo las soluciones a los subproblemas de $i < n$ hasta llegar a la solución del problema original con $i = n$. Esta técnica es justamente programación dinámica. Empleamos `memoization` guardando los resultados calculados previamente en un arreglo. El procedimiento explicado nos permite realizar una exploración implícita del espacio de soluciones. La solución final será óptima porque en el minuto `n` elegimos haber atacado hace `j` minutos, donde `j` maximiza la ecuación de recurrencia.