Exercici 3 (2.5 punts)

Una operació habitual als laboratoris de biologia molecular és la de trossejar una cadena d'ADN, tallant-la a unes determinades posicions d'interès. En termes computacionals, s'acostuma a representar una cadena d'ADN com un mot sobre l'alfabet $\{A, C, G, T\}$.

Assumim que tallar una cadena d'ADN $c = c_1 \cdots c_n$ de longitud n en dos trossos qualssevol té cost n. Quan parlem de fer un tall a la posició p_i, el símbol d'aquella posició queda com a últim símbol del tros esquerre (és a dir, tallem darrera de p_i).

És fàcil veure que, si es vol fer una seqüència de talls a unes determinades posicions d'interès $p_1, \ldots, p_k$, amb $p_i \in [1, \ldots, n]$, l'ordre en què es facin aquests talls afectarà el cost total de l'operació. Preneu com a exemple el cas d'haver de trencar una cadena de longitud $n = 200$ a les posicions 20, 80, 100 i 150. Si fem els talls d'esquerra a dreta, primer tallem una cadena de 200 (a la posició 20) i ens queden dos trossos, un de 20 i un altre de 180; el segon tall el farem a la posició 80 amb cost 180, etc. El cost total serà $600 = 200 + 180 + 120 + 100$. En canvi, podríem fer el primer tall a la posició 100 (amb cost 200) i després tallar cadascun dels dos trossos ($c_1 \cdots c_{100}$ i $c_{101} \cdots c_{200}$) d'esquerra a dreta. Això ens donaria un cost de $480 = 200 + (100 + 80) + (100)$.

Doneu un algorisme de PD tal que, donada una cadena d'ADN i la seqüència de posicions a l'esquerra de les quals volem tallar la cadena, ens proporcioni la forma menys costosa de tallar-la. El vostre algorisme ha de proporcionar el millor cost possible per realitzar els talls i una estructura de dades que ens permiti esbrinar en quin ordre procedir amb els talls per tal d'aconseguir el cost òptim.

Una solució:

[NOTA: Observeu que aquest problema assembla al problema de la multiplicació d'una cadena de matrius (vist a teoria), on s'ha de decidir com parentitzar les multiplicacions.]

Anomenem P al conjunt de k posicions d'interès que determinen on s'han de realitzar els talls de la seqüència. Assumirem que P no té repeticions (no té sentit que en tingui) i que està ordenat creixentment, és a dir,

$$P = \{p_1, \ldots, p_k\} \quad \text{on} \quad \forall i \in [1, k] : p_i \in [1, \ldots, n], \quad \text{i} \quad \forall i \in [1, k) : p_i < p_{i+1}.$$

Subestructura òptima: Identifiquem primer la subestructura òptima del problema. Considerem una seqüència òptima $p_{\pi(1)}, p_{\pi(2)}, \ldots, p_{\pi(k)}$ de realització dels talls, és a dir, la seqüència òptima en què primer es trenca la seqüència C a $c[p_{\pi(1)}]$, després a $c[p_{\pi(2)}]$, etc. Matemàticament, la funció $\pi : \{1, \ldots, k\} \to \{1, \ldots, k\}$ és una permutació de k elements que ens permet descriure una determinada permutació de la seqüència P de talls a realitzar.

Considerem el primer punt de tall $p_{\pi(1)}$, que trenca la cadena C en dues subcadenes C_1 i C_2.[1] Aleshores, la subestructura òptima és com segueix: de la subseqüència de talls

[1] Sigui $c_\alpha = c[p_{\pi(1)}]$, aleshores $C_1 = [c_1, \ldots, c_\alpha]$ i $C_2 = [c_\alpha + 1, \ldots, c_n]$.

© Professorat GRAD-A

$p_{\pi(2)}, p_{\pi(3)} \ldots, p_{\pi(k)}$ restants, els que són més petits que $p_{\pi(1)}$ trenquen C_1 de manera òptima, i els que són més grans que $p_{\pi(1)}$ trenquen C_2 de manera òptima.

Recurrència: Seguint la subestructura òptima descrita, definim $T(i, j)$ com el cost òptim de dur a terme el subconjunt ordenat de talls $\{p_i, \ldots, p_j\}$ de P.

Afegirem a P dos talls addicionals: $p_0 = 0$ i $p_{k+1} = n + 1$. Aquest afegitó tècnic no modifica en res la solució obtinguda i ens facilitarà la formulació de la recurrència. Aleshores, $T(i, j)$ es calcula segons indica la següent recurrència:

$\forall i, j \in [0, k + 1]$:

$$T(i,j) = \begin{cases} 0, & \text{si } j = i \text{ o } j = i+1 \text{ (casos base).} \\ \min\limits_{i<t<j} \{T(i,t) + T(t,j) + (p_j - p_i)\}, & \text{altrament.} \end{cases}$$

Observeu que els casos base els defineixen dues situacions: quan $j = i$ i quan $j = i+1$. Ambdues representen situacions en què no es realitza cap tall. En la resta de casos, el cost òptim de dur a terme els talls $\{p_i, \ldots, p_j\}$ requereix buscar el punt de tall $p_t \in \{p_i, \ldots, p_j\}$ que optimitza els costos dels subproblemes $\{p_i, \ldots, p_t\}$ i $\{p_t, \ldots, p_j\}$ generats, sumat al propi cost de fer un tall sobre el segment $\{p_i, \ldots, p_j\}$ (que sempre és el mateix, independentment del punt p_t on es triï fer el tall).

Segons aquesta definició, la solució a l'objectiu del problema la computa el valor $T(0, k + 1)$ d'aquesta recurrència.

Cost: L'entrada al problema és la seqüència $C = [c_1, \ldots, c_n]$ d'ADN i el conjunt $P = \{p_1, \ldots, p_k\}$ de talls a realitzar, on $k = |P| \leq |C| = n$. El cost temporal de l'algoritme ve determinat pels $\Theta(k^2)$ subproblemes que s'han de resoldre, i pel cost $\Theta(k)$ que necessita cadascun d'aquests subproblemes pel fet d'haver de cercar el punt de tall. Per tant, el cost temporal total és $\Theta(k^3)$. El cost espaial és $\Theta(k^2)$, necessari per tabular el $T(i, j)$ de cada subproblema.

Construcció de la solució òptima: La resolució de $T(0, k+1)$ ens permet obtenir el cost de la millor solució que particiona la seqüència. Si addicionalment volem també especificar quina és exactament la seqüència de talls que correspon a la solució òptima, necessitem guardar, per a cada $T(i, j)$, quin ha estat el punt de tall t que ha produït el mínim a la recurrència. Això és molt fàcil de implementar fent que cada posició de la taula sigui una tupla on es guardi també aquest valor.

Una vegada s'ha computat $T(0, k + 1)$, podem anar reconstruïnt la seqüència òptima de talls accedint recursivament als valors òptims corresponents a les dues solucions òptimes parcials (esquerra i dreta) que defineixen cada tall. Observeu que, en realitat, aquesta seqüència de talls ens està representant una mena d'arbre binari on es van veient els trossos que van produïnt cadascun dels talls segons l'ordre resultant. Si, per

© Professorat GRAUA

exemple, volem descriure com es van produïnt els talls per nivells, podríem implementar un algorisme com el següent:

```
funció IMPRIMEIX-TALLS-ÒPTIMS (T, k)

    Q.PUSH((0, k + 1))                ▷ T(0, k + 1) té la solució òptima al problema
    mentre no buida Q fer             ▷ Cua de trossos tallats
        (i, j) ← Q.POP()
        t ← T(i, j).SECOND()          ▷ Punt de tall òptim del tros en tractament
        PRINT(t)
        si t − i > 1 aleshores
            Q.PUSH((i, t))            ▷ Encuem el tros esquerre
        fi si
        si j − t > 1 aleshores
            Q.PUSH((t, j))            ▷ Encuem el tros dret
        fi si
    fi mentre
fi funció
```

Aquesta funció té cost $\Theta(k)$.

©Professorat GRAU-A