PAA - Aula ~~18~~ 22

Problemas da Mochila Inteira e do Alinhamento de Sequências

Princípios para projetar um algoritmo de programação dinâmica:

1. Imaginar uma solução ótima para o problema, com alguma estrutura/ordem.

2. Analisar esta solução genérica em busca de uma subestrutura ótima,
    - que dependa de um número pequeno de subproblemas.

3. Obter a recorrência a partir da demonstração da subestrutura ótima.

4. Derivar um algoritmo iterativo que, guiado pela recorrência,
    - preenche a tabela começando pelos subproblemas menores.

5. A solução final deve ser um valor da tabela (normalmente a última posição).

Eficiência do algoritmo vai depender do número total de subproblemas,
    - que determina o tamanho e dimensões da tabela,
- e do número de termos do qual depende a recorrência.

Problema da Mochila

Na entrada do problema da mochila inteira/binária

- recebemos n itens, sendo que cada item i, $1 \leq i \leq n$
  - tem peso $p_i > 0$ e valor $v_i > 0$
- Também recebemos uma capacidade $C \in \mathbb{Z}^+$

Uma solução para o problema é um subconjunto $S \subseteq \{1, \ldots, n\}$

- cujo peso total não ultrapassa C, i.e., $\sum_{i \in S} p_i \leq C$
  - e que maximize o valor total $\sum_{i \in S} v_i$

Quiz: verifiquem que o algoritmo guloso usado para a mochila fracionária,

- não devolve o ótimo para a seguinte entrada:

$n = 3 \quad C = 4$

$p_1 = 3 \quad v_1 = 7$

$p_2 = 2 \quad v_2 = 4$

$p_3 = 2 \quad v_3 = 4$

Vamos usar $I(n, C)$ para denotar uma entrada com n itens e capacidade C

- deixando implícito que cada item i tem valor v_i e peso p_i, para $1 \leq i \leq n$

Encontrando a subestrutura ótima

Considere uma entrada $I(n, C)$ e uma correspondente solução ótima S

- Vamos focar nossa atenção no último item n . Temos duas possibilidades:

  1. $n \notin S$
  2. $n \in S$

Primeiro, vamos analisar o Caso 1. Neste caso, S é uma solução ótima

- para $I(n-1, C)$, ou seja, para a mochila de mesma capacidade
  - que só pode usar os $n-1$ primeiros itens.

Prova: S é uma solução para $I(n-1, C)$, já que

  - ela não usa o item n e respeita a capacidade C
- Para mostrar que ela é ótima para $I(n-1, C)$ vamos supor, por contradição,
  - que existe uma solução S^* para $I(n-1, C)$, tal que $v(S^*) > v(S)$

- S^* é solução de $I(n, C)$.
- Mas $v(S^*) > v(S)$ que é ótimo p/ $I(n, C)$ (Contradição !!!)

Neste caso, qual o valor da solução ótima S

- em função da solução ótima dos subproblemas?

$v(S)$ = valor do ótimo p/ $I(n-1, C)$

Vamos analisar o Caso 2, em que n está em S. Como queremos descrever S
- com soluções ótimas de subproblemas, vamos considerar $S' = S \setminus \{n\}$

- Temos que S' só usa itens dentre os $n-1$ primeiros. Além disso,
  - como S respeita a capacidade C, i.e., $\sum_{i \in S} p_i \leq C$
- ao remover o n de S temos que S' respeita uma restrição mais forte

$$\sum_{i \in S'} p_i = \sum_{i \in S} p_i - p_n \leq C - p_n$$

- Assim, S' é uma solução ótima para $I(n-1, C-p_n)$, ou seja, para a mochila
  - de capacidade $C-p_n$ que só pode usar os $n-1$ primeiros itens.

Prova: S' é uma solução para $I(n-1, C-p_n)$, pois S'
- não usa o item n e respeita a capacidade $C-p_n$
- Para mostrar que ela é ótima para $I(n-1, C-p_n)$ vamos supor,
  - por contradição, que existe S^* melhor que S' para $I(n-1, C-p_n)$
- Note que $S^* \cup \{n\}$ é uma solução para $I(n, C)$, já que
  - respeita a capacidade C e só usa itens válidos para esta entrada.
- Como $v(S^* \cup \{n\}) = v(S^*) + v_n > v(S') + v_n = v(S)$
  - temos uma contradição, já que S era ótima para $I(n, C)$

Neste caso, qual o valor de S em função da solução ótima dos subproblemas?

$v(S) =$ valor do ótimo p/ $I(n-1, C-p_n) + v_n$

Escrevendo a recorrência

Nossos subproblemas dependem

- tanto dos itens que podem ser usados na solução
  - quanto da capacidade residual da mochila.

Por isso, nossa recorrência terá dois parâmetros, i e x

- sendo que $A[i,x]$ é o valor de uma solução ótima para uma mochila que
  - só pode ser preenchida usando os i primeiros itens, $i = 0, \ldots, n$
    - e tem capacidade x, $x = 0, \ldots, C$

Seguindo a análise da subestrutura ótima, temos a seguinte recorrência

$$A[i,x] = \max \{ A[i-1,x], v_i + A[i-1, x-p_i] \}$$

1° caso | 2° caso

Observe que esta recorrência tem uma limitação, pois

- o segundo caso não faz sentido quando $p_i > x$
  - já que a capacidade da mochila fica negativa.
- Além disso, os casos bases da recorrência ocorrem
  - quando o número de itens disponível é 0
- Neste caso, a mochila sempre terá valor 0
  - independente da capacidade.

Projetando o algoritmo de Programação Dinâmica

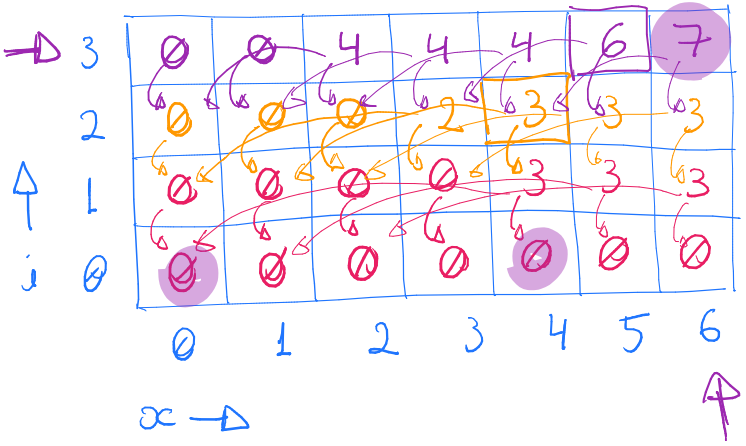
Exemplo de preenchimento da tabela:

$n = 3 \qquad C = 6$

$v_1 = 3 \qquad p_1 = 4$

$v_2 = 2 \qquad p_2 = 3$

$v_3 = 4 \qquad p_3 = 2$

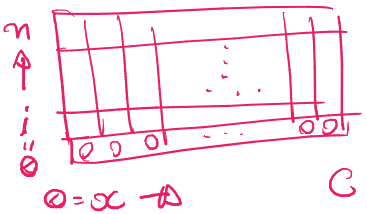


$$A[i,x] = \max\{\underbrace{A[i-1,x]}_{1^{\circ}\text{ caso}}, \underbrace{v_i + A[i-1, x-p_i]}_{2^{\circ}\text{ caso}}\}$$

Pseudocódigo do algoritmo:

```
mochila(n, v, p, C):
    para x = 0 até C:
        A[0,x] = 0
    para i = 1 até n:
        para x = 0 até C:
            se x < p_i: A[i,x] = A[i-1,x]
            senão:
                A[i,x] = max{A[i-1,x], v_i + A[i-1,x-p_i]}
    devolva A[n,C]
```

$\Theta(n)$ $O(C)$ $O(1)$



Prova de corretude: por indução matemática usando

- como hipótese de indução a recorrência que demonstramos.

Eficiência: $\Theta(n \cdot C)$, o que faz deste algoritmo pseudopolinomial, já que

- o número C pode crescer como uma exponencial no tamanho da entrada.

Reconstruir a solução usa as mesmas ideias apresentadas na última aula.

Problema do Alinhamento de Sequências (ou da Distância de Edição)

Entrada: duas sequências $X = x_1 x_2 x_3 \dots x_m$ e $Y = y_1 y_2 \dots y_n$
- ambas definidas sobre um mesmo alfabeto σ
- Penalidades $\alpha(a,b) \geq 0$ paga por casar os símbolos a e b de σ
  - Em geral, se $a = b$ então $\alpha(a,b) = 0$
- Penalidade $\alpha_{gap} \geq 0$ paga por casar um símbolo com um espaço.

Solução: um alinhamento de custo mínimo.
- Um alinhamento é uma modificação das sequências originais
  - que pode inserir espaços entre símbolos,
    - mas não pode inverter ou remover quaisquer símbolos.

Exemplo 1: X = A G G G C T e Y = A G G C A.
- Neste caso, $m = 6$, $n = 5$, $\sigma = \{A, G, C, T\}$
- Uma solução possível é

A G G G C T
A G G - C A

  - e seu custo é $\alpha_{gap} + \alpha(T, A)$

Note que uma solução pode envolver adição de espaços em ambas as sequências.

Exemplo 2: X = G G T C C e Y = A G G C C.

- Neste caso, $m = 5$, $n = 5$, $\sigma = \{A, G, C, T\}$
- Uma solução possível é

-	G	G	T	C	C
A	G	G	-	C	C

- e seu custo é $2\alpha_{gap}$

Vamos usar $I(m,n)$ para denotar uma entrada do problema principal

- para o problema do Alinhamento de Sequências em que:
  - a primeira sequência (X) tem m itens
  - e a segunda sequência (Y) tem n itens.
- Deixamos implícitas as penalidades α

Vamos usar o símbolo $\&$ para denotar um alinhamento, ou seja,

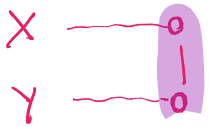
- $X \& Y$ indica um alinhamento das sequências X e Y
- Exatamente qual é esse alinhamento será definido pelo contexto.

Encontrando a subestrutura ótima

Considere uma entrada $I(m,n)$ e um correspondente alinhamento ótimo X & Y

- Focando na última posição deste alinhamento, temos três possibilidades:
  1. x_m alinhado com y_n
  2. x_m alinhado com —
  3. y_n alinhado com —

- Não consideramos a opção espaço alinhado com espaço, pois este caso
  - se reduz a uma solução melhor (removendo a última posição).
- Sejam $X' = X - x_m$ e $Y' = Y - y_n$



Primeiro, vamos analisar o Caso 1. Neste caso o alinhamento X' & Y'

- que obtemos ao remover a última posição do alinhamento ótimo X & Y
  - é uma solução ótima para o subproblema $I(m-1, n-1)$

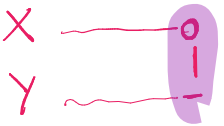
Prova: sendo P o custo do nosso alinhamento de X' & Y', por contradição,

- suponha que existe um alinhamento mais barato de X' e Y' com custo P^*
  - Quiz3: chegue num absurdo para completar a prova.

No Caso 1, qual o valor da solução ótima em função do ótimo dos subproblemas?

$\alpha(X[m] \& Y[n]) = \alpha(X'[m-1] \& Y'[n-1]) + \alpha(x_m, y_n)$

Vamos analisar o Caso 2, em que x_m é alinhado com espaço



- Neste caso o alinhamento $X'[m-1] \& Y[n]$ obtido ao remover
  - a última posição do alinhamento ótimo $X[m] \& Y[n]$
    - é uma solução ótima para o subproblema $I(m-1, n)$

Prova: sendo P o custo do nosso alinhamento de $X'[m-1] \& Y[n]$, por contradição,

- suponha que existe um alinhamento melhor de $X'[m-1]$ e $Y[n]$ com custo P^*
  - Quiz4: chegue num absurdo para completar a prova.

No Caso 2, qual o valor da solução ótima em função do ótimo dos subproblemas?

$\alpha(X[m] \& Y[n]) = \alpha(X'[m-1] \& Y[n]) + \alpha_{gap}$

A análise do Caso 3 é idêntica à do Caso 2, exceto por trocar X' por Y' e Y por X

Escrevendo a recorrência

Nossos subproblemas dependem tanto do tamanho da primeira sequência,

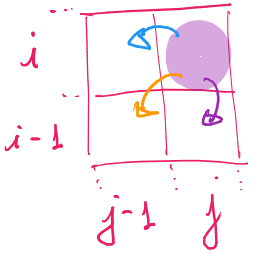
- quanto do tamanho da segunda sequência.

Por isso, nossa recorrência terá dois parâmetros, i e j

- sendo que $A[i,j]$ é o valor de uma solução ótima em que
  - a primeira sequência é o prefixo de tamanho i de X, $i = 0, \ldots, m$
  - e a segunda sequência é o prefixo de tamanho j de Y, $j = 0, \ldots, n$

Seguindo nossa análise da subestrutura ótima, temos a seguinte recorrência

$$A[i,j] = \min \{ A[i-1, j-1] + \alpha(x_i, y_j), A[i-1, j] + \alpha_{gap}, A[i, j-1] + \alpha_{gap} \}$$



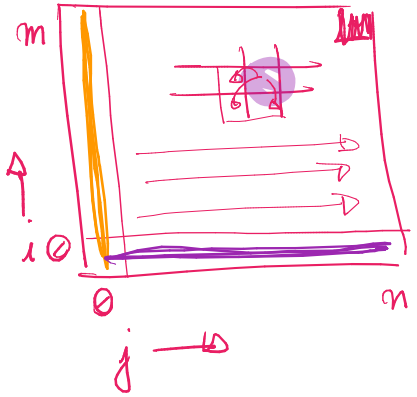
Observe que os casos base dessa recorrência ocorrem quando $i = 0$ ou $j = 0$

- Nestes casos, a sequência cujo prefixo não é nulo
  - terá de ser alinhada com espaços. Portanto,

$$A[i,0] = i\,\alpha_{gap}, \quad 0 \leq i \leq m$$

$$A[0,j] = j \cdot \alpha_{gap}, \quad 0 \leq j \leq n$$

Projetando o algoritmo de Programação Dinâmica



Pseudocódigo do algoritmo:

```
alinhamentoSeq(X, m, Y, n, α):
O(m) — para i = 0 até m: A[i,0] = i α_gap
O(n) — para j = 0 até n: A[0,j] = j α_gap
para i = 1 até m:                                          } Θ(m.n)
    para j = 1 até n:                                      }
        A[i,j] = min{ A[i-1,j-1] + α(X_i, Y_j),            } O(1)
                      A[i-1,j] + α_gap, A[i,j-1] + α_gap }
devolva A[m,n]
```

Prova de corretude: por indução matemática usando

- como hipótese de indução a recorrência que demonstramos antes.

Eficiência: $\Theta(m.n)$, por conta dos dois laços aninhados

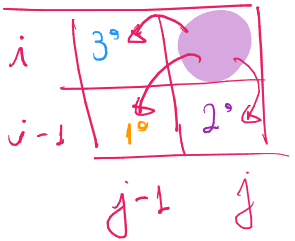
- e do fato que a recorrência pode ser resolvida em tempo constante.

Reconstruindo a solução: dada uma tabela A preenchida pelo algoritmo,

- começamos na posição $A[m,n]$ e vamos voltando de acordo
  - com o caso da recorrência usado para preencher a posição atual.

```
rec Sol(X, m, Y, n, α, A):
    i = m, j = n
    enquanto i ≥ 1 e j ≥ 1:
        se A[i,j] == A[i-1, j-1] + α(X_i, Y_j): // 1°
            X_i & Y_j, i--, j--
        senão se A[i,j] == A[i-1, j] + α_gap: // 2°
            X_i & -, i--
        senão // 3°: A[i,j] == A[i, j-1] + α_gap
            - & Y_j, j--
    enquanto i > 0: X_i & -, i--
    enquanto j > 0: - & Y_j, j--
    devolva o alinhamento obtido
```

$O(n+m)$



Eficiência: $\Theta(m+n)$, pois em cada iteração do laço principal

- pelo menos i ou j são diminuídos de uma unidade.