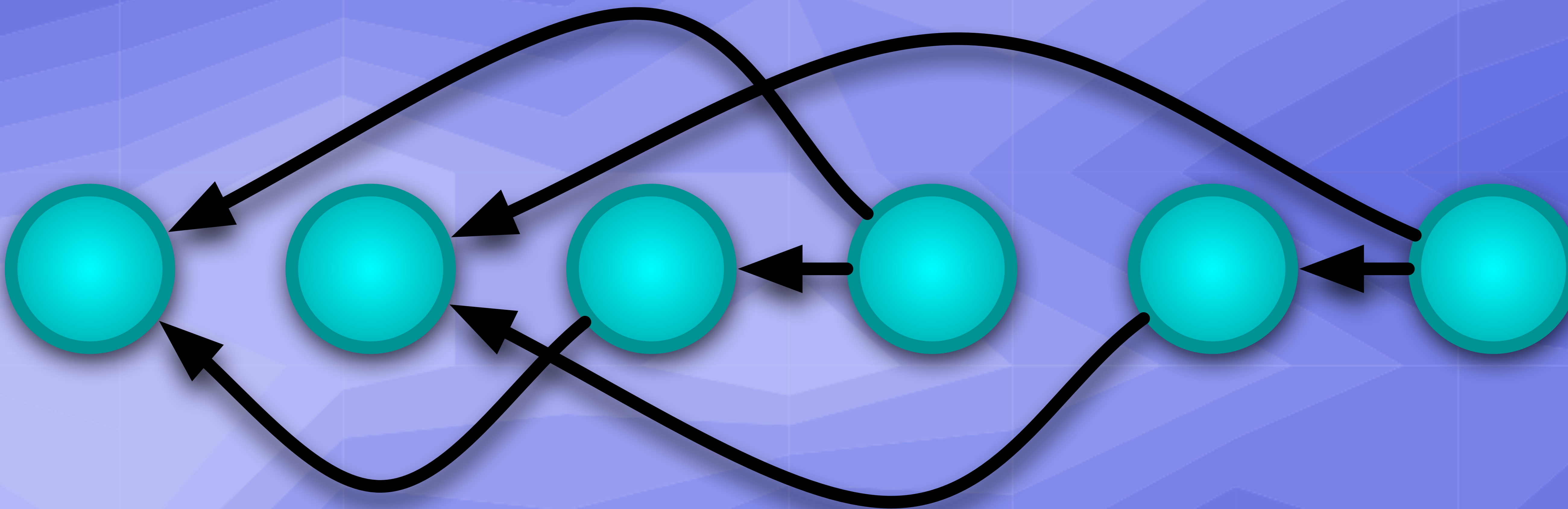


Programação Dinâmica II

SCC0210 — Algoritmos Avançados (2/2011)

Lucas Schmidt Cavalcante

- Longest Increasing Sequence (LIS) $O(n^2)$
- Longest Increasing Sequence (LIS) $O(n \log n)$

Longest Increasing Subsequence (LIS) O(n²)

Dada uma sequência de números inteiros (*v[]*), desejamos saber a maior subsequência crescente.

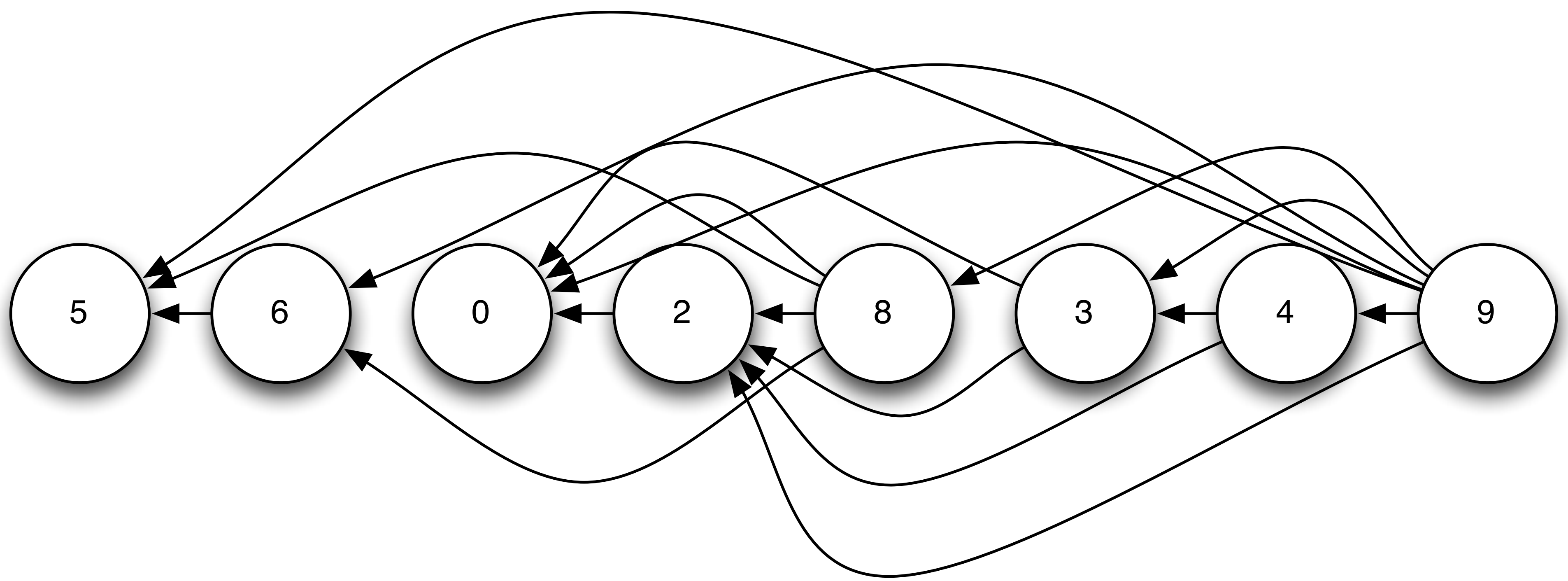
5	6	0	2	8	3	4	9

Uma possível solução de tamanho 4 é 5-6-8-9, mas a maior tem tamanho 5 (0-2-3-4-9).

Longest Increasing Subsequence (LIS) O(n^2)

Vamos pensar em Grafo...

5	6	0	2	8	3	4	9



Adicione uma aresta do elemento i até todo elemento j ($j < i$) que for menor do que o elemento i. Depois basta achar o maior caminho. Funciona, mas gasta muito espaço além de ser custovo para adicionar todas as arestas e achar o maior caminho.

Longest Increasing Sequence (LIS) O(n²)

Dada uma sequência de números inteiros (*v[]*), desejamos saber a maior subsequência crescente.

5	6	0	2	8	3	4	9

Uma possível solução de tamanho 4 é 5-6-8-9, mas a maior tem tamanho 5 (0-2-3-4-9).

Caso base: um número forma uma sequência de tamanho um.

Solução trivial para o primeiro (5) e o menor (0) elemento.

Estado: no vetor *pd[]*, a posição *i* guarda o tamanho da maior sequência que contém o elemento *i*.

Longest Increasing Sequence (LIS) $O(n^2)$

Dada uma sequência de números inteiros (*v[]*), desejamos saber a maior subsequência crescente.

5	6	0	2	8	3	4	9

Uma possível solução de tamanho 4 é 5-6-8-9, mas a maior tem tamanho 5 (0-2-3-4-9).

Caso base: um número forma uma sequência de tamanho um.

Solução trivial para o primeiro (5) e o menor (0) elemento.

Estado: no vetor *pd[]*, a posição *i* guarda o tamanho da maior sequência que contém o elemento *i*.

Então a cada iteração basta saber se o elemento *i* extende uma subsequência: $pd[i] = max(pd[j] \text{ tal que } v[j] < v[i]), 0 \leq j \leq i$.

Longest Increasing Sequence (LIS) O(n²)

Ao analisar um novo elemento *i* da sequência, verifique se ele extende uma subsequência: $pd[i] = max(pd[j] \text{ tal que } v[j] < v[i]), 0 \leq j \leq i$.

5	6	0	2	8	3	4	9

1							

Primeiro elemento não tem nenhuma subsequência para extender, então forma uma composta por ele mesmo.

Longest Increasing Sequence (LIS) O(n^2)

Ao analisar um novo elemento *i* da sequência, verifique se ele extende uma subsequência: $pd[i] = max(pd[j] \text{ tal que } v[j] < v[i]), 0 \leq j \leq i$.

5	6	0	2	8	3	4	9

1	2						

6 pode criar uma subsequência de tamanho 1, mas pode expandir a subsequência de tamanho 1 de 5.

Longest Increasing Sequence (LIS) $O(n^2)$

Ao analisar um novo elemento *i* da sequência, verifique se ele extende uma subsequência: $pd[i] = max(pd[j] \text{ tal que } v[j] < v[i]), 0 \leq j \leq i$.

5	6	0	2	8	3	4	9

1	2	1					

0 não consegue expandir nenhuma subsequência já encontrada, pois é menor do que 5 ou 6, então forma sua própria.

Longest Increasing Sequence (LIS) O(n^2)

Ao analisar um novo elemento *i* da sequência, verifique se ele extende uma subsequência: $pd[i] = max(pd[j] \text{ tal que } v[j] < v[i]), 0 \leq j \leq i$.

5	6	0	2	8	3	4	9

1	2	1	2				

2 só é maior que 0, então escolher entre criar sua própria de tamanho 1 ou expandir a do 0 e criar uma de tamanho 2.

Longest Increasing Sequence (LIS) $O(n^2)$

Ao analisar um novo elemento *i* da sequência, verifique se ele extende uma subsequência: $pd[i] = max(pd[j] \text{ tal que } v[j] < v[i]), 0 \leq j \leq i$.

5	6	0	2	8	3	4	9

1	2	1	2	3			

8 é o maior número até agora, então pode expandir qualquer uma das subsequências. Não faz sentido expandir uma sequência de tamanho 1, então pode expandir de 6 ou de 2 (tanto faz).

Longest Increasing Sequence (LIS) $O(n^2)$

Ao analisar um novo elemento *i* da sequência, verifique se ele extende uma subsequência: $pd[i] = max(pd[j] \text{ tal que } v[j] < v[i]), 0 \leq j \leq i.$

5	6	0	2	8	3	4	9

1	2	1	2	3	3	4	5

Ao final se for preciso saber a subsequência, use um vetor extra para marcar de onde veio.

A cada número tem que olhar todos atrás, logo, complexidade $O(n^2)$.

Longest Increasing Subsequence O(nlogn)

A estratégia $O(n^2)$ era de saber se o elemento *i* extende alguma subsequência $j < i$.

Note que só era importante saber a maior subsequência menor que *i.*

- De fato, interessa saber todas as subsequências de tamanho *j*? Não, só importa aquela de menor valor, ou seja, a mais "expandível".

Longest Increasing Subsequence O(nlogn)

Resolvendo o LIS por manter somente o menor elemento que forma uma subsequência de tamanho *j*.

5	6	0	2	8	3	4	9

5							

Mais uma vez 5 é caso base e forma uma sequência de tamanho 1 (na primeira posição de pd[]).

Longest Increasing Subsequence O(nlogn)

Resolvendo o LIS por manter somente o menor elemento que forma uma subsequência de tamanho *j*.

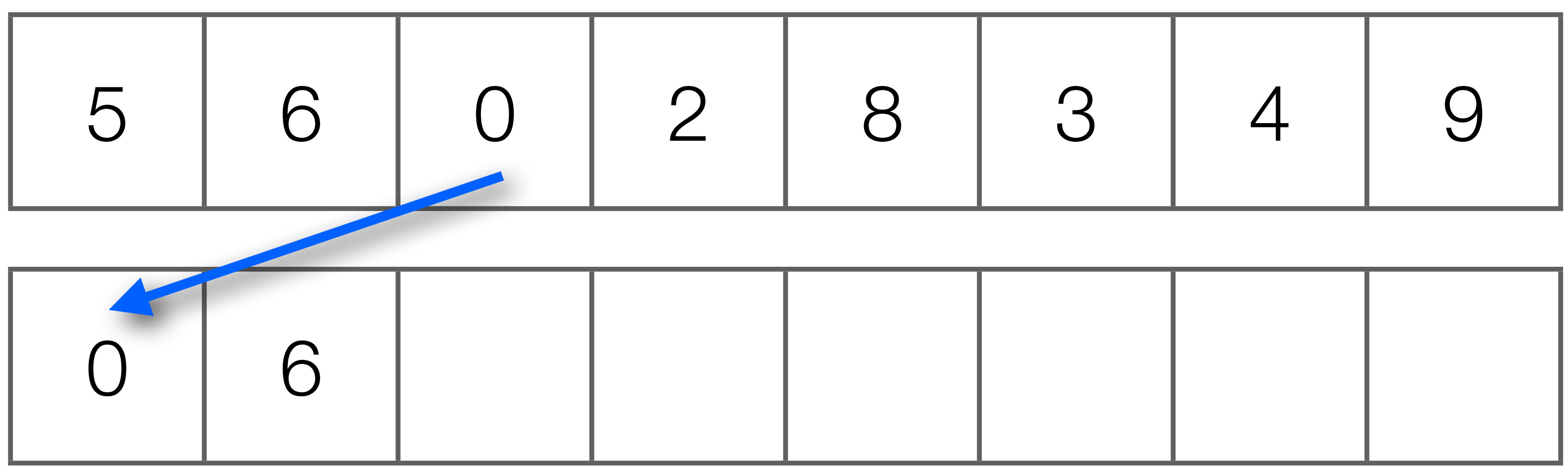
5	6	0	2	8	3	4	9

5	6						

6 é maior do que 5, logo pode expandir a subsequência de tamanho 1 para uma de 2.

Longest Increasing Subsequence O(nlogn)

Resolvendo o LIS por manter somente o menor elemento que forma uma subsequência de tamanho *j*.



0 é menor que 5 e 6, então não expande nenhuma subsequência, mas é capaz de criar uma subsequência de tamanho 1 “melhor”.

Longest Increasing Subsequence O(nlogn)

Resolvendo o LIS por manter somente o menor elemento que forma uma subsequência de tamanho *j*.

5	6	0	2	8	3	4	9

0	2						

2 é menor que 6 mas é maior que 0, então só pode expandir a subsequência de tamanho 1. A subsequência com 2 é melhor do que a anterior, então substitui.

Longest Increasing Subsequence O(nlogn)

Resolvendo o LIS por manter somente o menor elemento que forma uma subsequência de tamanho *j*.

5	6	0	2	8	3	4	9

0	2	8					

8 é maior que 0 mas não forma uma subsequência de tamanho 2 melhor, no entanto pode expandir para uma de 3 (maior que 2).

Longest Increasing Subsequence O(nlogn)

Resolvendo o LIS por manter somente o menor elemento que forma uma subsequência de tamanho *j*.

5	6	0	2	8	3	4	9

0	2	3	4				

4 consegue expandir a maior subsequência atual, então basta criar uma nova de tamanho 4.

Longest Increasing Subsequence O(nlogn)

Resolvendo o LIS por manter somente o menor elemento que forma uma subsequência de tamanho *j*.

5	6	0	2	8	3	4	9

0	2	3	4	9			

9 consegue expandir a maior subsequência atual, então basta criar uma nova de tamanho 5.

Longest Increasing Subsequence O(nlogn)

Resolvendo o LIS por manter somente o menor elemento que forma uma subsequência de tamanho *j*.

5	6	0	2	8	3	4	9

0	2	3	4	9			

Note que o vetor pd[] será sempre crescente, ou seja, pd[1] < pd[2] < ... < pd[j].

Para cada elemento *i* precisamos achar a primeira posição em que ele é maior. Use busca binária e a complexidade fica em O(nlogn).

Referências

- Longest Increasing Sequence, Wikipedia.
- Longest Increasing Sequence, Algorithmist.
- Dynamic Programming Practice Problems.
- "*Introduction to Algorithms: A Creative Approach*", Udi Manber. Addison-Wesley.
- "*Competitive Programming: Increasing the Lower Bound of Programming Contests*", Steven Halim & Felix Halim.