動的計画法：DP

北海道大学 情報科学研究科

情報知識ネットワーク研究室 修士1年

栗田 和宏

DPって何？

- DP(Dynamic Programming)とはある計算式に対して，一度計算した結果を記憶しておき，効率化を図ることである．
- 記憶するメモリのことをDPテーブルと呼ぶ

具体例：フィボナッチ数

- DPを使ってフィボナッチ数を求めてみよう！！

具体例：フィボナッチ数

- fibonacci(5) → fibonacci(4) + fibonacci(3)

具体例：フィボナッチ数

- fibonacci(5) → fibonacci(4) + fibonacci(3)
- fibonacci(4) → *fibonacci(3) + fibonacci(2)

具体例：フィボナッチ数

- fibonacci(5) → fibonacci(4) + fibonacci(3)
- fibonacci(4) → *fibonacci(3) + fibonacci(2)
- fibonacci(3) → *fibonacci(2) + fibonacci(1)

具体例：フィボナッチ数

- fibonacci(5) → fibonacci(4) + fibonacci(3)
- fibonacci(4) → *fibonacci(3) + fibonacci(2)
- fibonacci(3) → *fibonacci(2) + fibonacci(1)
- fibonacci(2) → *fibonacci(1) + fibonacci(0)

具体例：フィボナッチ数

- fibonacci(5) → fibonacci(4) + fibonacci(3)
- fibonacci(4) → *fibonacci(3) + fibonacci(2)
- fibonacci(3) → *fibonacci(2) + fibonacci(1)
- fibonacci(2) → *fibonacci(1) + fibonacci(0)
- *fibonacci(1) → 1
  …

具体例：フィボナッチ数

1, 1, ?, ?, ?, ?

具体例：フィボナッチ数

1, 1, 2, ?, ?, ?

具体例：フィボナッチ数

1, 1, 2, 3, ?, ?

具体例：フィボナッチ数

1, 1, 2, 3, 5, ?

具体例：フィボナッチ数

1, 1, 2, 3, 5, 8

具体例：フィボナッチ数

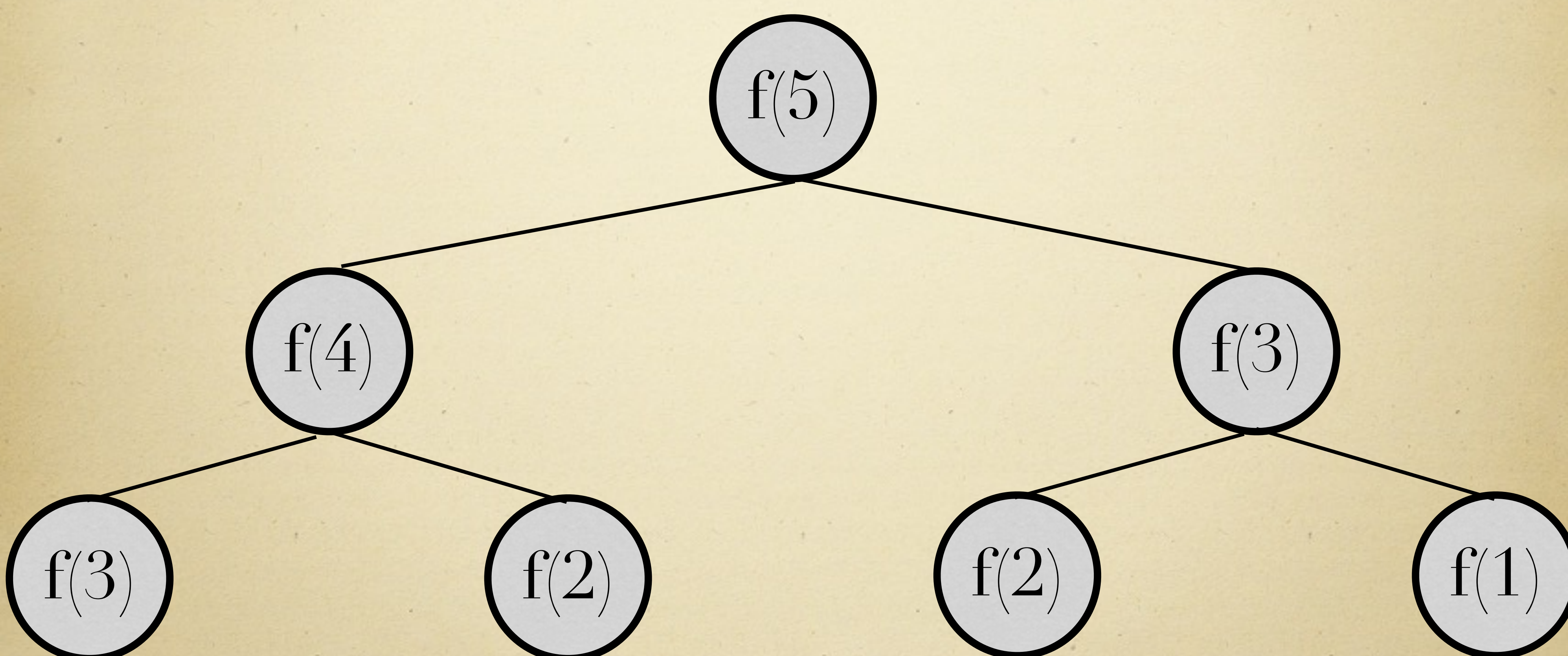
- 愚直なアルゴリズム

  計算量：$O(2^n)$

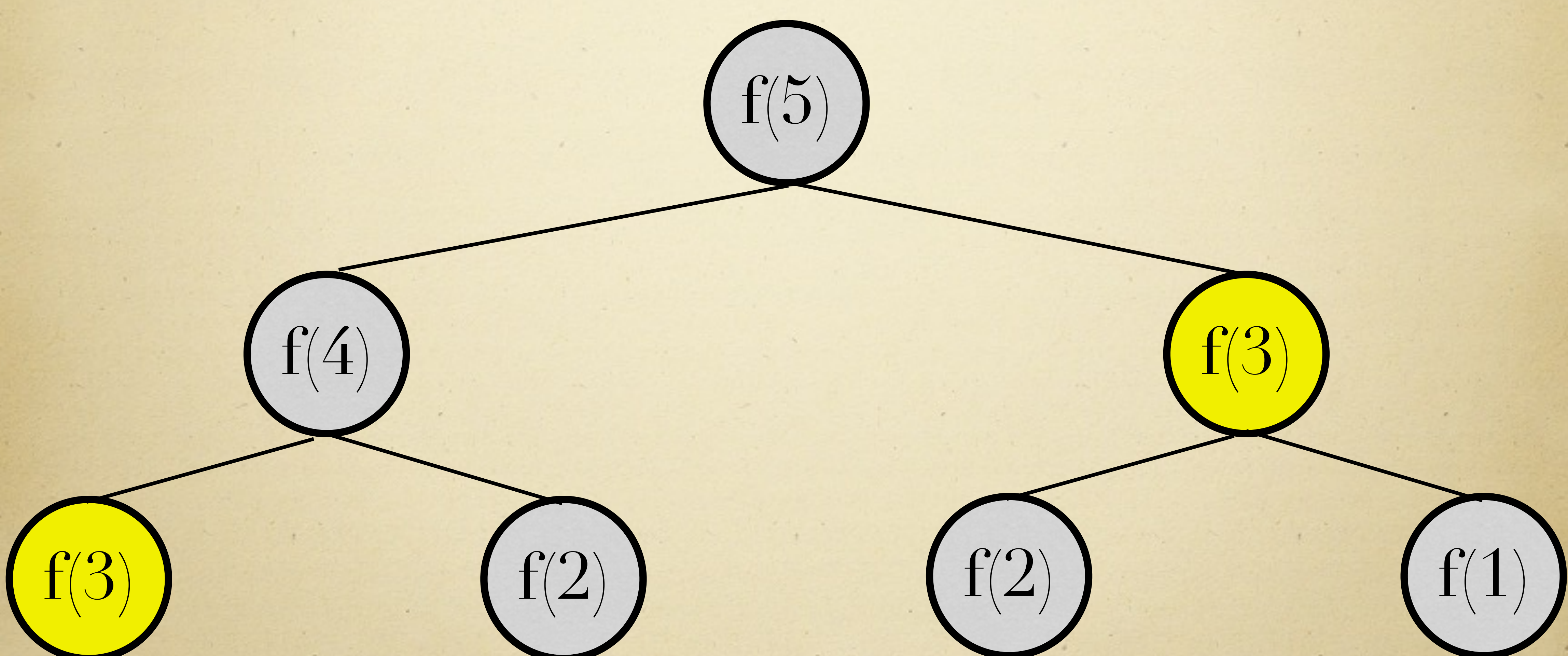
- DPを用いたアルゴリズム

  計算量：$O(n)$

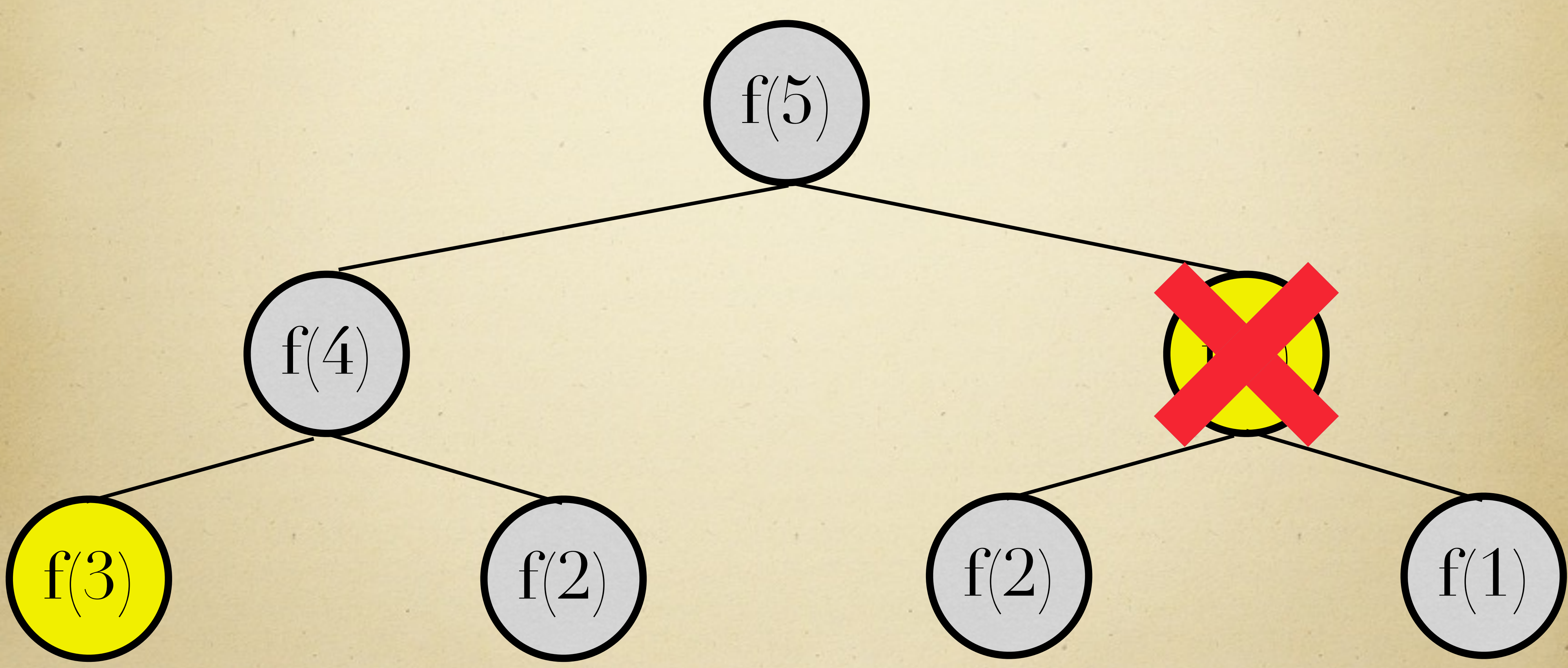
なぜ計算量が落ちたか？



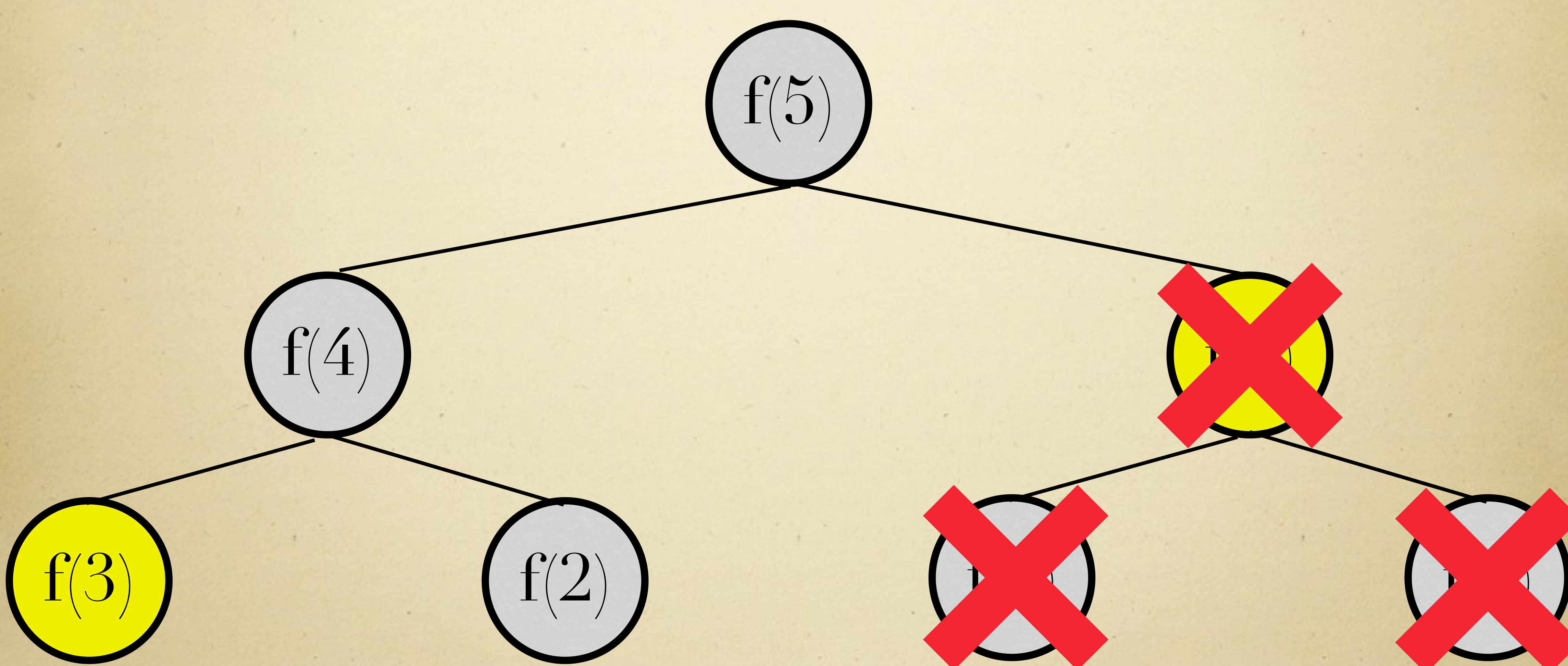
なぜ計算量が落ちたか？



なぜ計算量が落ちたか？



なぜ計算量が落ちたか？



ソースコードの例

- https://github.com/kazu0423/procon_example/blob/master/fibonacci_dp.cpp

最長共通部分列

- **最長共通部分列**(Longest Common Subsequence)とは2つの与えられた列XとYの最長の共通部分列を求める問題である.

部分列とは？

- 列X = abcの部分列

  {emp, a, b, c, ab, bc, ac, abc}　(emp:空集合)

- 大雑把に言うと元の文字列の順番は変えずに,

  幾つか文字を取ってきてできる列

愚直な解法

- 列Xと列Yの部分列を列挙する.
- 列Xの1つの部分列と列Yの全ての部分列を比較して共通部分の長さを求める
- 最長なものが見つかる

$n = \max(|X|, |Y|)$

愚直な解法

- 列Xと列Yの部分列を列挙する． → $O(2^n)$
- 列Xの1つの部分列と列Yの全ての部分列を比較して共通部分の長さを求める → $O(n^3)$
- 最長なものが見つかる → $O(1)$

$n = \max(|X|, |Y|)$

自明な考察

- 例えば列Xと列YのLCSがわかっていると仮定する.
- このとき，X’ = X + aと Y’ = Y + aのLCSは？

自明な考察

- 例えば列Xと列YのLCSがわかっていると仮定する.
- このとき，X’ = X + aとY’ = Y + aのLCSは？
- 当然，列X’と列Y’とのLCS + 1である.

記憶する計算

- 列X_iと列Y_jのLCSがわかっていれば，列X_{i+1}と列Y_{j+1}のLCSをO(1)求められる．

記憶する計算

- 列Xの先頭からi個の列をX_iとする.
- 列X_iと列Y_jのLCSがわかっていれば，列X_{i+1}と列Y_{j+1}のLCSをO(1)求められる.
- ということは列X_iと列Y_jのLCSを記憶しておけば良い.

DPテーブル

X = abacd, Y = abcded

X\Y	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0						
2	0						
3	0						
4	0						
5	0						

DPテーブル

X = abacd, Y = abcded

X\Y	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	1	1	1	1	1
2	0						
3	0						
4	0						
5	0						

DPテーブル

X = abacd, Y = abcded

X\Y	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	1	1	1	1	1
2	0	1	2	2	2	2	2
3	0						
4	0						
5	0						

DPテーブル

X = abacd, Y = abcded

X\Y	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	1	1	1	1	1
2	0	1	2	2	2	2	2
3	0	1	2	2	2	2	2
4	0	1	2	3	3	3	3
5	0	1	2	3	4	4	4

DPテーブル

X = abacd, Y = abcded

X\Y	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	1	1	1	1	1
2	0	1	2	2	2	2	2
3	0	1	2	2	2	2	2
4	0	1	2				
5	0						

DPテーブル

X = abacd, Y = abcded

X\Y	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	1	1	1	1	1
2	0	1	2	2	2	2	2
3	0	1	2	2	2	2	2
4	0	1	2				
5	0						

DPテーブル

X = abacd, Y = abcded

X\Y	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	1	1	1	1	1
2	0	1	2	2	2	2	2
3	0	1	2	2	2	2	2
4	0	1	2	3			
5	0						

DPテーブル

X = abacd, Y = abcded

X\Y	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	1	1	1	1	1
2	0	1	2	2	2	2	2
3	0	1	2	2	2	2	2
4	0	1	2	3			
5	0						

DPテーブル

X = abacd, Y = abcded

X\Y	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	1	1	1	1	1
2	0	1	2	2	2	2	2
3	0	1	2	2	2	2	2
4	0	1	2	3			
5	0						

DPテーブル

X = abacd, Y = abcded

X\Y	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	1	1	1	1	1
2	0	1	2	2	2	2	2
3	0	1	2	2	2	2	2
4	0	1	2	3	3		
5	0						

更新の計算量

- DPテーブルの各マスはO(1)で更新可能.
- マスの数は|X| * |Y|
- 計算量はO(|X| * |Y|)になる.
- 発展問題としてLongest common substringという問題もある.

ソースコードの例

- https://github.com/kazu0423/procon_example/blob/master/longest_common_subsequence.cpp

連鎖行列積

- n個の行列の連鎖M_1，M_2，…，M_nが与えられたとき，乗算回数の最小を求めよ

- 制約

  $1 < n, r, c < 100$

問題例

乗算回数：0回

3
5
8

5	9	2

1	6	3
7	2	4
6	4	8

問題例

乗算回数：9回

15	27	6
25	45	10
40	72	16

1	6	3
7	2	4
6	4	8

問題例

乗算回数：36回

15	27	6
25	[illegible]	[illegible]
40	72	16

1	6	3
[illegible]	[illegible]	4
6	4	8

掛け算!!

問題例

乗算回数：0回

3
5
8

5	9	2

1	6	3
7	2	4
6	4	8

問題例

乗算回数：9回

3
5
8

80	56	67

問題例

乗算回数：18回

掛け算!!

発展：連鎖行列積

- 行列の乗算の順番によって乗算回数が大きく異なる.
- 乗算回数の最小値はいくつか？

愚直な解放

- やっぱりまずは全探索.
  これで解ければ一番楽!!

愚直な解放

- やっぱりまずは全探索.
  これで解ければ一番楽!!

- しかし全探索は$O(n!)$かかる → nの最大値は100
  これはまずい

愚直な解放

- やっぱりまずは全探索.
  これで解ければ一番楽!!
- しかし全探索はO(n!)かかる → nの最大値は100
  これはまずい
- DPを使おう

何を記憶するか？

- 行列M_1とM_2を考える．M_1は(c_1, r_1)，M_2は(c_2, r_2)の行列である．

- これらの行列の乗算回数はr_1*c_1*r_2である．

何を記憶するか？

- 行列M_1とM_2を考える．M_1は(c_1, r_1)，M_2は(c_2, r_2)の行列である．
- これらの行列の乗算回数は$r_1*c_1*r_2$である．
- しかし，我々が知りたいのは$M_1M_2M_3 \ldots M_n$の乗算回数である．

何を記憶するか？

- ここで，$M'_1 = M_1 \ldots M_k$ と $M'_2 = M_{k+1} \ldots M_m$ に分けて考える
- もし，M'_1 と M'_2 を作る最小の乗算回数がわかれば $M'_1 M'_2$ を作る最小の乗算回数がわかる．

何を記憶するか？

- ここで，$M'_1 = M_1 \ldots M_k$ と $M'_2 = M_{k+1} \ldots M_m$ に分けて考える
- もし，M'_1 と M'_2 を作る最小の乗算回数がわかれば $M'_1 M'_2$ を作る最小の乗算回数がわかる．
- 覚えておこう

何を記憶するか？

- M’$_1$とM’$_2$の作り方はO(m)通りなので，O(m)通りの中の最小値を求める．
- まとめるとDPテーブルにはi番目からj番目までの最小の乗算回数を記憶させることになる．

DPテーブル

$M_1 M_2 M_3 M_4 M_5 M_6$　簡単のためすべて2*2行列

i\j	1	2	3	4	5	6
1	0	inf	inf	inf	inf	inf
2	emp	0	inf	inf	inf	inf
3	emp	emp	0	inf	inf	inf
4	emp	emp	emp	0	inf	inf
5	emp	emp	emp	emp	0	inf
6	emp	emp	emp	emp	emp	0

DPテーブル

$M_1M_2M_3M_4M_5M_6$　簡単のためすべて2*2行列

i\j	1	2	3	4	5	6
1	0	inf	inf	inf	inf	inf
2	emp	0	inf	inf	inf	inf
3	emp	emp	0	inf	inf	inf
4	emp	emp	emp	0	inf	inf
5	emp	emp	emp	emp	0	inf
6	emp	emp	emp	emp	emp	0

DPテーブル

$M_1 M_2 M_3 M_4 M_5 M_6$　簡単のためすべて2*2行列

i\j	1	2	3	4	5	6
1	0	8	inf	inf	inf	inf
2	emp	0	inf	inf	inf	inf
3	emp	emp	0	inf	inf	inf
4	emp	emp	emp	0	inf	inf
5	emp	emp	emp	emp	0	inf
6	emp	emp	emp	emp	emp	0

DPテーブル

$M_1 M_2 M_3 M_4 M_5 M_6$　　簡単のためすべて2*2行列

i\j	1	2	3	4	5	6
1	0	8	inf	inf	inf	inf
2	emp	0	inf	inf	inf	inf
3	emp	emp	0	inf	inf	inf
4	emp	emp	emp	0	inf	inf
5	emp	emp	emp	emp	0	inf
6	emp	emp	emp	emp	emp	0

DPテーブル

$M_1 M_2 M_3 M_4 M_5 M_6$　　簡単のためすべて2*2行列

i\j	1	2	3	4	5	6
1	0	8	inf	inf	inf	inf
2	emp	0	$\boxed{8}$	inf	inf	inf
3	emp	emp	0	inf	inf	inf
4	emp	emp	emp	0	inf	inf
5	emp	emp	emp	emp	0	inf
6	emp	emp	emp	emp	emp	0

DPテーブル

$M_1 M_2 M_3 M_4 M_5 M_6$　簡単のためすべて2*2行列

i\j	1	2	3	4	5	6
1	0	8	inf	inf	inf	inf
2	emp	0	8	inf	inf	inf
3	emp	emp	0	inf	inf	inf
4	emp	emp	emp	0	inf	inf
5	emp	emp	emp	emp	0	inf
6	emp	emp	emp	emp	emp	0

DPテーブル

$M_1 M_2 M_3 M_4 M_5 M_6$ 簡単のためすべて2*2行列

i\j	1	2	3	4	5	6
1	0	8	inf	inf	inf	inf
2	emp	0	8	inf	inf	inf
3	emp	emp	0	8	inf	inf
4	emp	emp	emp	0	inf	inf
5	emp	emp	emp	emp	0	inf
6	emp	emp	emp	emp	emp	0

DPテーブル

$M_1 M_2 M_3 M_4 M_5 M_6$　簡単のためすべて2*2行列

i\j	1	2	3	4	5	6
1	0	8	inf	inf	inf	inf
2	emp	0	8	inf	inf	inf
3	emp	emp	0	8	inf	inf
4	emp	emp	emp	0	8	inf
5	emp	emp	emp	emp	0	8
6	emp	emp	emp	emp	emp	0

DPテーブル

$M_1M_2\,M_3\,M_4\,M_5\,M_6$　　簡単のためすべて2*2行列

i\j	1	2	3	4	5	6
1	0	8	inf	inf	inf	inf
2	emp	0	8	inf	inf	inf
3	emp	emp	0	8	inf	inf
4	emp	emp	emp	0	8	inf
5	emp	emp	emp	emp	0	8
6	emp	emp	emp	emp	emp	0

DPテーブル

$M_1 M_2 M_3 M_4 M_5 M_6$　簡単のためすべて2*2行列

i\j	1	2	3	4	5	6
1	0	8	inf	inf	inf	inf
2	emp	0	8	inf	inf	inf
3	emp	emp	0	8	inf	inf
4	emp	emp	emp	0	8	inf
5	emp	emp	emp	emp	0	8
6	emp	emp	emp	emp	emp	0

DPテーブル

$M_1 M_2 M_3 M_4 M_5 M_6$　簡単のためすべて2*2行列

i\j	1	2	3	4	5	6
1	0	8	16	inf	inf	inf
2	emp	0	8	inf	inf	inf
3	emp	emp	0	8	inf	inf
4	emp	emp	emp	0	8	inf
5	emp	emp	emp	emp	0	8
6	emp	emp	emp	emp	emp	0

DPテーブル

$M_1 M_2 M_3 M_4 M_5 M_6$　簡単のためすべて2*2行列

i\j	1	2	3	4	5	6
1	0	8	16	inf	inf	inf
2	emp	0	8	16	inf	inf
3	emp	emp	0	8	16	inf
4	emp	emp	emp	0	8	16
5	emp	emp	emp	emp	0	8
6	emp	emp	emp	emp	emp	0

DPテーブル

$M_1 M_2 M_3 M_4 M_5 M_6$　簡単のためすべて2*2行列

i\j	1	2	3	4	5	6
1	0	8	16	inf	inf	inf
2	emp	0	8	16	inf	inf
3	emp	emp	0	8	16	inf
4	emp	emp	emp	0	8	16
5	emp	emp	emp	emp	0	8
6	emp	emp	emp	emp	emp	0

DPテーブル

$M_1 M_2 M_3 M_4 M_5 M_6$　簡単のためすべて2*2行列

i\j	1	2	3	4	5	6
1	0	8	16	inf	inf	inf
2	emp	0	8	16	inf	inf
3	emp	emp	0	8	16	inf
4	emp	emp	emp	0	8	16
5	emp	emp	emp	emp	0	8
6	emp	emp	emp	emp	emp	0

DPテーブル

$M_1 M_2 M_3 M_4 M_5 M_6$　簡単のためすべて2*2行列

i\j	1	2	3	4	5	6
1	0	8	16	inf	inf	inf
2	emp	0	8	16	inf	inf
3	emp	emp	0	8	16	inf
4	emp	emp	emp	0	8	16
5	emp	emp	emp	emp	0	8
6	emp	emp	emp	emp	emp	0

DPテーブル

$M_1 M_2 M_3 M_4 M_5 M_6$　簡単のためすべて2*2行列

i\j	1	2	3	4	5	6
1	0	8	16	inf	inf	inf
2	emp	0	8	16	inf	inf
3	emp	emp	0	8	16	inf
4	emp	emp	emp	0	8	16
5	emp	emp	emp	emp	0	8
6	emp	emp	emp	emp	emp	0

DPテーブル

$M_1 M_2 M_3 M_4 M_5 M_6$　簡単のためすべて2*2行列

i\j	1	2	3	4	5	6
1	0	8	16	24	inf	inf
2	emp	0	8	16	inf	inf
3	emp	emp	0	8	16	inf
4	emp	emp	emp	0	8	16
5	emp	emp	emp	emp	0	8
6	emp	emp	emp	emp	emp	0

DPテーブル

$M_1 M_2 M_3 M_4 M_5 M_6$　簡単のためすべて2*2行列

i\j	1	2	3	4	5	6
1	0	8	16	24	32	40
2	emp	0	8	16	24	32
3	emp	emp	0	8	16	24
4	emp	emp	emp	0	8	16
5	emp	emp	emp	emp	0	8
6	emp	emp	emp	emp	emp	0

更新の計算量

- 大雑把に考えると一つのマスを更新するのに最悪でも$O(n)$時間で更新できる.
- マスの数はn^2なので更新には$O(n^3)$時間がかかる.

更新の計算量

- 大雑把に考えると一つのマスを更新するのに最悪でも$O(n)$時間で更新できる.
- マスの数はn^2なので更新には$O(n^3)$時間がかかる.
- これならnが100でも余裕で間に合う

ソースコードの例

- https://github.com/kazu0423/procon_example/blob/master/matrix_chain_multiplication_problem.cpp

まとめ

- DPとは途中結果を記憶して同じ計算をしないようにするテクニック

- DPの計算量を出すときは
  マスの数 * 1マスを埋めるのにかかる時間

- 他にも有名な問題としてナップサック問題というのもある.