Dynamic Programming: same as divide & conquer except that we "store" some of the recursive function calls.

$F_0 = F_1 = 1$: $F_n = F_{n-1} + F_{n-2}$.

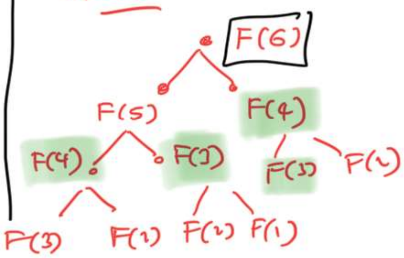
1, 1, 2, 3, 5, 8, 13, 21, 34, ...

```
F(n) {
    if n=0 or n=1
        return 1
    else
        return F(n-1) + F(n-2)
}.
```

the same recursive call is being made multiple times.

running time is exponential in n.

n = 20.

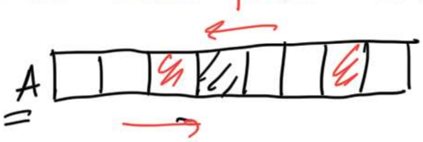


1. Divide & Conquer Alg. : the running time is high because we are making the same recursive calls again and again

2. What are all the possible recursive calls?

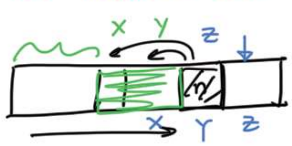
   $F(k)$, $k \le n$.

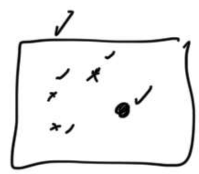
   store the values for them in a DP Table.



$A[k] = F(k)$

3. Compute the entries of A in a manner such that when we want to compute entries in the order such that no recursive call is needed.





$A[n] = A[n-1] + A[n-2]$ if $n \geq 3$.

$= 1$ $n = 1, 2$.

$A[1] = A[2] = 1$.

for $i = 3 \ldots n$

~~$A[i] = \underbrace{A[i-1] + A[i-2]}$.~~

$z = y + x$

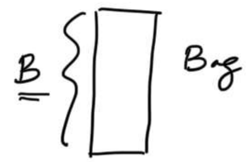
save space! $O(1)$ space.

$x = y$
$y = z$

② Knapsack Problem:

items — $1, \dots, n$ (indivisible)

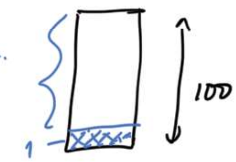
item i: size s_i, profit p_i



Problem: Find a maximum profit subset of items which fit in the bag.

p_i / s_i

2 items
size: 1, 2 — profit
100, 100 | 2
100.



Divide & Conquer Alg.

subset of items. bag cap set of items.

```
int Knapsack( B, 1, 2, ..., n ) {
    if n=1 ....

    return max{ Knapsack( B - s_1, 2, ..., n ) + p_1
                Knapsack( B, 2, ..., n ) }
}
```

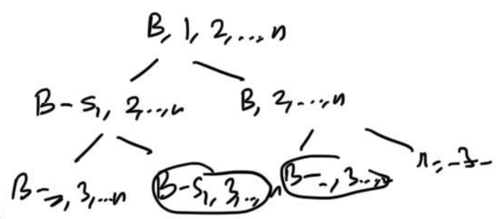
← if we choose to select item 1.

item 1 — take this item. / not take this item.

What are the possible recursive calls?

$(b,\ i, \dots, n)$ $\quad \boxed{Bn}$

$0 \le b \le B$

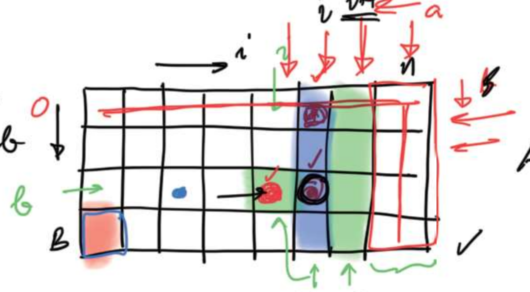


Store all of these recursive calls in a table

$O(nB)$

$O(nB)$ time

for $b = 0$ to B
  for $i = n-1$ downto 1



for $i = n-1$ to 1
  for $b = 0$ to B
  ⊛

$O(B)$. space.

$A[b, i]$ will store the max. profit that we can get when bag has size b and the items are $i, \dots, n$

for $i = 0 \dots n$
  for $b = 0 \dots B$ ✗

$A[B, 1]$

only if $b \ge s_i$

$$A[b,i] = \max\left(A[b-s_i, i+1] + p_i,\ A[b, i+1]\right) \quad (*)$$

Knapsack $(b, \underline{i}, \ldots, n)$ {

$$\max \begin{cases} \text{Knapsack}(b - s_i,\ i+1, \ldots, n) + p_i & \text{only if } b \geq s_i \\ \text{Knapsack}(b,\ i+1, \ldots, n) \end{cases}$$

}.

\# bits to write down the input

$x^2 \quad x^3 \ldots$

is this polynomial time?

NP Complete.

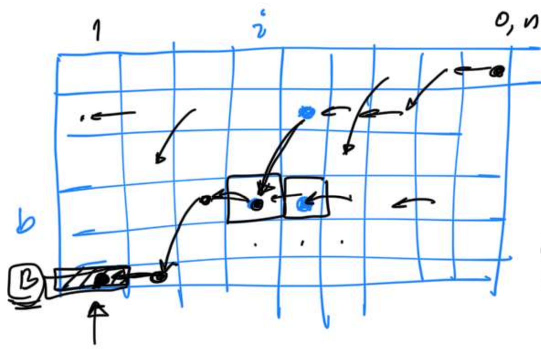
1. Is this an efficient algorithm? $O(nB)$. : exponential time alg.

$n \log B$

2. What if we also want to find out the actual set of items to pick?

$B = 10^{200} + 1$

201 bits.

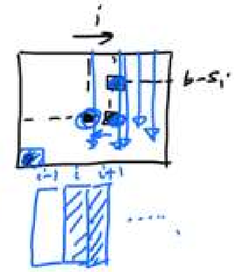


$O(nB)$

only if $b \geq s_i$

$A[b,i] = \max\left(A[b-s_i, i+1] + p_i,\ A[b, i+1] \right)$ (*)

items 1…n   items i+1,…,n



$A[B, 0]$

storage $O(Bn) \to O(B)$

$B = n$.

Knapsack (b, i,…, n) {

2^n n^2

max { Knapsack (b − s_i , i+1,…, n) + p_i — only if $b \geq s_i$
Knapsack (b, i+1,…, n)

}.

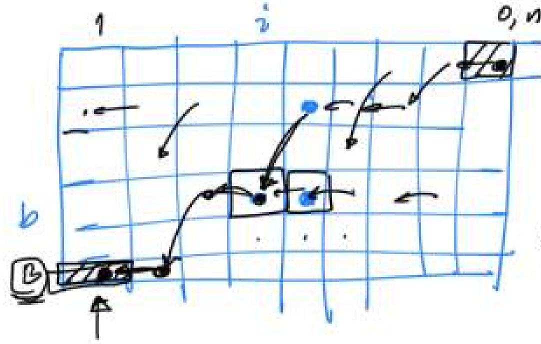
bits to write down the input
x^2 x^3 …

is this polynomial time?

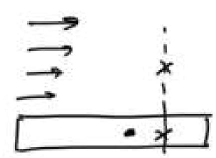
1) Is this an efficient algorithm? — NP Complete.
$O(nB)$: exponential time alg.
$n \log B$

2) What if we also want to find out the actual set of items to pick?

$B = 10^{200} + 1$
201 bits.

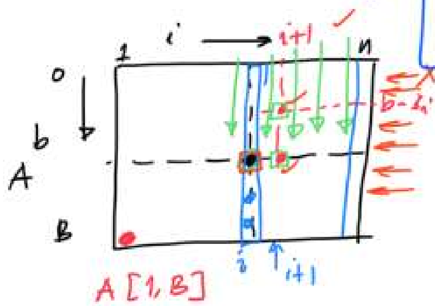


$O(nB)$



21/10/21. Knapsack

for i = n−1 down to 1
for b = 0 to B
(*)



$A[i;b]$ = max. profit
i, i+1, …, n
using a knapsack of capacity B.

1) Save space: to compute entries in the i^{th} column, we only need entries in the $(i+1)^{th}$ col.
O(B) space.

$A[i;b] = \max\left(\underbrace{A[i+1, b]}_{\text{don't take } i},\ \underbrace{A[i+1, b-s_i] + p_i}_{\text{take } i,\ b \geq s_i} \right)$ * O(1)

$\parallel$ O(Bn).

$B = n^2$

1, 2, …, n $B = 2^{\log B}$

|| polynomial time: poly. in the

S_i, P_i $\quad B$

$$\boxed{\sum_{i=1}^{n} (\underbrace{\log S_i + \log P_i}_{\geq n}) + \log B}$$

|| "size" of the input.
#bits needed to write down the input

(2) what if we also want to find the actual solution.
- also store the solution at each table entry: $O(n)$. $\Rightarrow O(nB)$ space.

Aside: n. $\quad k=2$

$(\log_2 n)$ $\quad O(n)$.

exponential time?

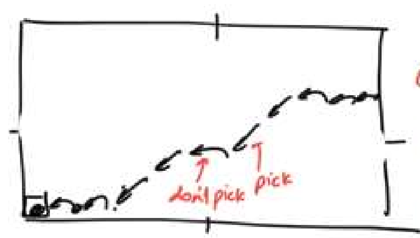
|| while (k < n)
  if (k divides) output not prime
  else k++.
output prime

$n : \log_2 n$ $\quad 10^{200}$ $\quad 10^{200}+1$

$100 \dots 01$

800

$A[1, B] = \max(\boxed{A[2,B]}, A[2, B-S_i] + P_i)$.



$O(nB)$ $\quad A[i,b]$.

(i) $A[i,b] = A[i+1, b]$

(ii) $A[i,b] = A[i+1, b-s_i] + p_i$

(i) $O(B)$ space if we just want the max. profit.

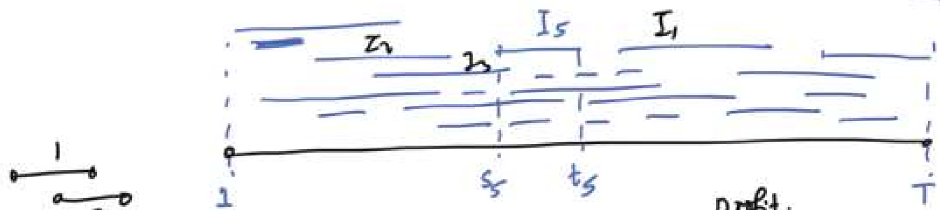
(ii) $O(nB)$ space if we want the actual solution.

(iii) $O(nB)$ time.

$\boxed{Q}$ Can we get $O(nB)$ time, $O(B)$ space and get the actual solution??

YES.

(2) Max. profit Interval Selection Problem:



$I_1, \dots, I_n$

$I_i = [s_i, t_i]$.

P_i: profit of interval I_i

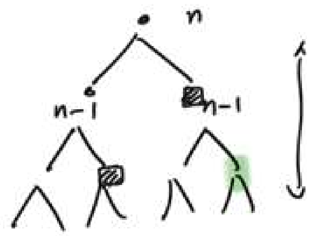
Pick a subset of intervals of max ~~size~~ profit which do not overlap (pairwise disjoint) (pairwise disjoint).

select($I_1, \ldots, I_n$) { $\underline{n=1}$: output I_1. (base case)

max (select($I_2, \ldots, I_n$), p_1 + select($\underbrace{\qquad\qquad}$)).

all intervals among $I_2, \ldots, I_n$ which do not overlap with I_1

}.



2^n running time.

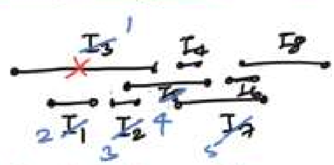
Dynamic Program: What are all possible recursive calls??

Select (any subset). ← 2^n

$I_k, \ldots, I_n$

$I_1, I_2, \ldots, I_n$

~~$t_1 \le t_2 \le t_3 \le \ldots \le t_n$~~



$s_1 \le s_2 \le s_3 \le \ldots \le s_3$

Select ($I_1, \ldots, I_n$) {

max (select($I_2, \ldots, I_n$),

p_1 + select( )

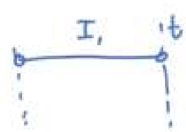
$I_2, I_4, I_5, \ldots, I_8$

select ($I_1, \ldots, I_n$) {

max(select ($I_2, \ldots, I_n$), p_1 + $\boxed{\text{select(intervals starting after } t_1 \text{)}}$

$I_k, \ldots, I_n$.

$s_i \ge t_1$



$I_1, I_2, \ldots, I_n$.

possible recursive calls : n.

select($I_\ell, I_{\ell+1}, \ldots, I_n$)

$A[n] = p_n$.

$O(n)$

$A[k] :=$ max profit when the intervals are $I_k, \ldots, I_n$

for $k = n-1$ downto 1

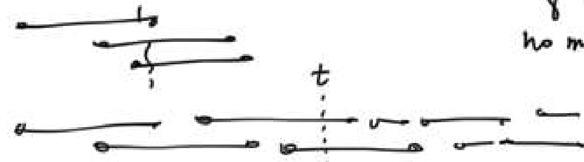
$A[k] =$ max ($A[k+1]$, $A[\ell] + p_k$)

I_ℓ is the first interval starting after t_k.

s_i

$O(n \cdot \log n)$.

we have two halls instead of one hall! Find the max. profit subset of intervals such that at any time t no more than 2 such intervals contain t.

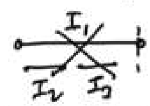


Exercise: Give a dynamic progr. alg. for this problem.

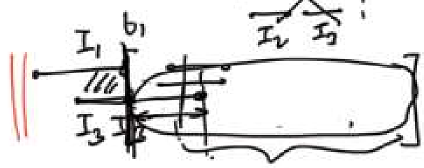
✓ $\boxed{\text{Select } (I_1, I_2, \ldots, I_n)}$ { $s_1 \le s_2 \le \ldots \le s_n$.

don't select I_1 select$(I_2, \ldots, I_n)$

$(\overset{b}{=}, I_i, \ldots, I_n)$ then I_1:



$I_k, \ldots, I_n$



---

(3) You own two shops: A, B: $A \rightleftarrows B$ costs C amt of money.

	1,	2,	...,	n
$\boxed{A}$:	p_1^A	p_2^A	...	p_n^A
B	p_1^B	p_2^B	...	p_n^B

p_i^A: amount of money earned if present at A on day i

p_i^B

On day 1, A. A, A, B, B, B, A, A, A, Max. [total profit − Cost].

−C −C

MaxProfit $(1, 2, \ldots, n)$ {

(A, B)

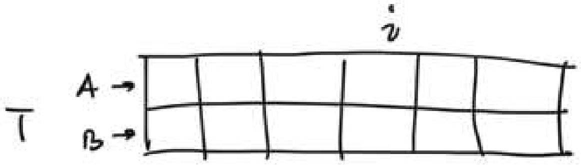
$p_1^A + \underline{\text{MaxProfit}(2, \ldots, n)}$

2n.

MaxProfit $(A, i, \ldots, n)$ {

$p_i^A + \max\Big[\text{maxprofit}(A, i+1, \ldots, n),$
$\text{maxprofit}(B, i+1, \ldots, n) - C\Big]$

}.



$T[A, i]$: max profit $i, \ldots, n$ if we start at A on day i

T[A,i] : max ...

T[B,i] :

---

(4) Edit Distance Problem: edit: (i) Add a character (ii) Delete a char.

Given two strings $\boxed{s_1, \ldots, s_n}$ and $\boxed{t_1, \ldots, t_m}$

define EDIT DISTANCE between them is defined as

the min # of edits in the first string to get to the second string.

$\boxed{ABABA} \rightarrow ABAB \rightarrow A\,A\,B \rightarrow ACAB.$

ACAB

Another way of defining: we only insert characters in both the strings so that they are identical

Edit distance = # stars getting added.

$\left\{\begin{array}{llllll} A & B\,C & A & B & \cancel{A} \\ A & \boxed{B}\,\boxed{C} & A & B & \boxed{A} \end{array}\right.$ $\Big\|$ $\begin{array}{cccccc} A & B & \boxed{*} & A & B & \boxed{A} \\ A & * & C & A & B & * \end{array}$

$\quad\quad 1 \ldots i \quad\quad 1 \ldots j$

EditDistance ((A[1...n]), (B[1..m])) { Base Case...

Min (ⓐ EditDistance (A[1... n-1], B[1...,m]) + 1, ... A[n] ... * ‖ ⓐ

or

ⓑ EditDistance (A[1.. n], B[1... m-1]) + 1 ... * ... B[m] ‖ ⓑ

or

ⓒ Edit (A[1...n-1], B[1... m-1]) ... A[n] ... B[m] ⓒ

← only applies if A[n] = B[m]

}.

exponential time.

n, m

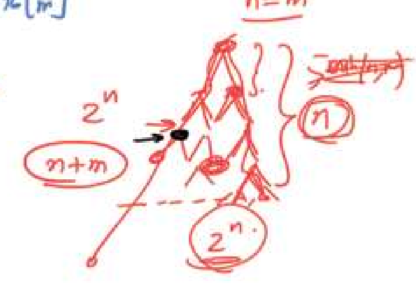
n-1, m n, m-1

n, 1

n + m

2^n

n=m



What are the different recursive calls?

(A[1..i], B[1-j]) ✓

≤ nm ✓

Store all of the recursive calls.

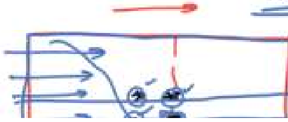
T(i,j): store EditDistance

j → nm

i ↓

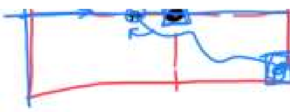
O(m)

T



$T(i,j)$: store Each Distance

$(A[1\ldots i], B[1-j])$



$T(0,j) = j$
$T(i,0) = i$

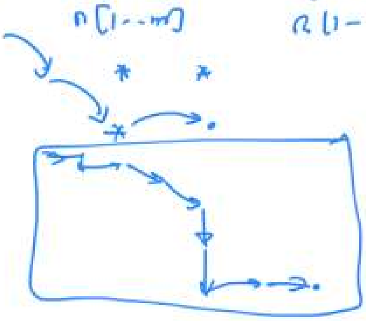
* $T(i,j) = \min\big(T(i,j-1)+1,\ T(i-1,j)+1,\ T(i-1,j-1) \big)$ only if $A[i] = B[j]$.

$i=0,\ j=0$

for $i = 1, \ldots, n$
  for $j = 1, \ldots, m$

$O(nm)$ time.

$A[1..n]$ $\quad A[1-i]$
$B[1..m]$ $\quad B[1-j]$



i, j, k

Maximum Flows:

electrical current, water, traffic....

directed graph

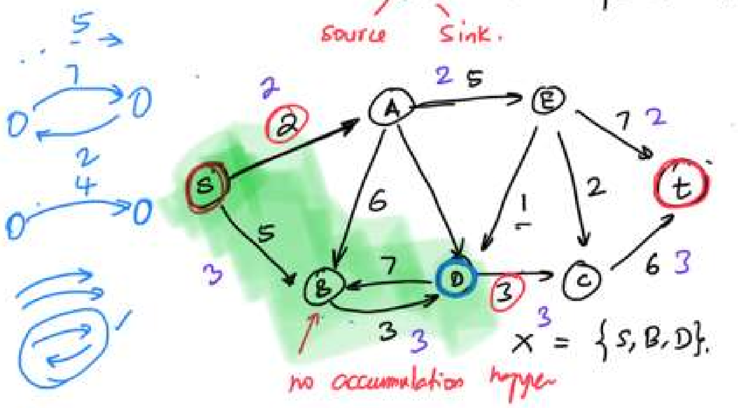
Problem : G, e has a capacity $u_e \geq 0$

s, t : two special vertices

source sink.

- Greedy
- Divide & Conquer
- Dynamic Programming
- Maximum Flows.

Convention: No edge enters s,
Notation: No edge leaves t.



$X^3 = \{s, B, D\}$.

- Electric current:
  each edge is a wire
  u_e : max. current that can flow on e.
- Water pipes:
  each edge is a water pipe
  u_e: rate at which water can flow on this pipe.
- traffic: each each is a road.
  u_e: how much traffic flow on each edge
  (rate of traffic)
  eg, 10 trucks / hour.

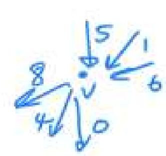
What do we want?

Defn: A flow f specifies a quantity f_e for each edge :

(i) $0 \leq f_e \leq u_e \quad \forall$ edge e.

(ii) "flow conservation"
for every vertex v other than s or t,

$$\sum_{e\,:\,e \text{ comes into } v} f_e = \sum_{e\,:\,e \text{ goes out of } v} f_e$$



$\delta^-(v)$: edges entering v.
$\delta^+(v)$: edges leaving v.

7 8 4

4