5112

9-19

Dynamic Programming

Subset-Sum

Knapsack

} Section 6.4

Tidying up WIS

```
M-Opt-Val(j)
  If j = 0
    Return 0
  Else if M[j] ≠ -1
    Return M[j]
  Else
    Set M[j] = max(M-Opt-Val(p(j)) + v_j, M-Opt-Val(j-1))
    Return M[j]
```

Subtle point: What does this function compute/output?

It's only the value.

What if we want to output the subset as well?

1. Could memoize the optimal subset solutions as we go.

j	OptVal	Subset
0	0	$\emptyset$
1	2	$\{1\}$
2	3.5	$\{2\}$
3	3.5	$\{2\}$
	.	
	.	
	.	
n	11.7	$\{n-1, n-2, \ldots, 7, 2, 1\}$

Does it work? yes

Running time? $O(n^2)$ naively

Append $\emptyset$ to $SS\{n-1\}$ can get back to $O(n)$

2. Compute the subset using the table of optimal values.

j	OptVal
0	0
1	2
2	3.5
3	3.5
	.
p(n)	5
n-1	11.7
n	11.7

```
OptSubset(j)
  If j == 0
    Return ∅
  Else
    If v_j + M[p(j)] ≥ M[j-1]
      Return OptSubset(p(j)) ∪ {j}
    Else
      Return OptSubset(j-1)
```

Append in O(1)

At most n recursions

$\Rightarrow O(n)$

Subset Sum

Input: $\{w_i\}$ $1 \le i \le n$. Threshold T.

Output: $S \subseteq \{1, \ldots, n\}$ such that

$$\sum_{i \in S} w_i \le T.$$

and we want this to be maximal.

Return S such that the weight of S is as close to T as possible without going over.

Subset Sum Greedy Algorithms

1. Highest Weight

$\{T/2+1, T/2, T/2\}$ —HW→ $\{T/2+1\}$

Not optimal!

$\{T/2+1, T/2, T/2\}$ —Opt→ $\{T/2, T/2\}$

2. Lowest Weight

$\{1, T/2, T/2\}$ —LW→ $\{1, T/2\}$

Not optimal

$\{1, T/2, T/2\}$ —Opt→ $\{T/2, T/2\}$

Subset Sum Dynamic Programming

Input: $\{w_1, \ldots, w_n\}$ T

Suppose Θ is an optimal subset

Is $n \in \Theta$?

Case 1: $n \notin \Theta$ $\Rightarrow$ $\Theta \subseteq \{w_1, \ldots, w_{n-1}\}$
$\Rightarrow$ Θ is an optimal subset of the problem on $\{w_1, \ldots, w_{n-1}\}$ w/ T

Case 2: $n \in \Theta$ what does $\Theta \setminus \{n\}$ look like?
$\Rightarrow$ $\Theta \setminus \{n\}$ has weight at most $T - w_n$
So $\Theta \setminus \{n\}$ is an optimal subset of the problem on $\{w_1, \ldots, w_{n-1}\}$ w/ threshold $T - w_n$

Subset Sum Dynamic Programming

Write $Opt(j, V)$ for weight of the solution to the problem on $\{w_1, \ldots, w_j\}$ with threshold V.

Lemma:

$$Opt(j, V) = \max_{\leq T}\left(Opt(j-1, V), Opt(j-1, V-w_j) + w_j\right)$$

Subset Sum

SubsetSum($\{w_1, \ldots, w_n\}$, T)

Array $M[0 \ldots n, 0 \ldots T]$

Set $M[0, t] = 0$ for $t = 0, 1, \ldots, T$

For $i = 1, 2, \ldots, n$

  For $t = 0, \ldots, T$

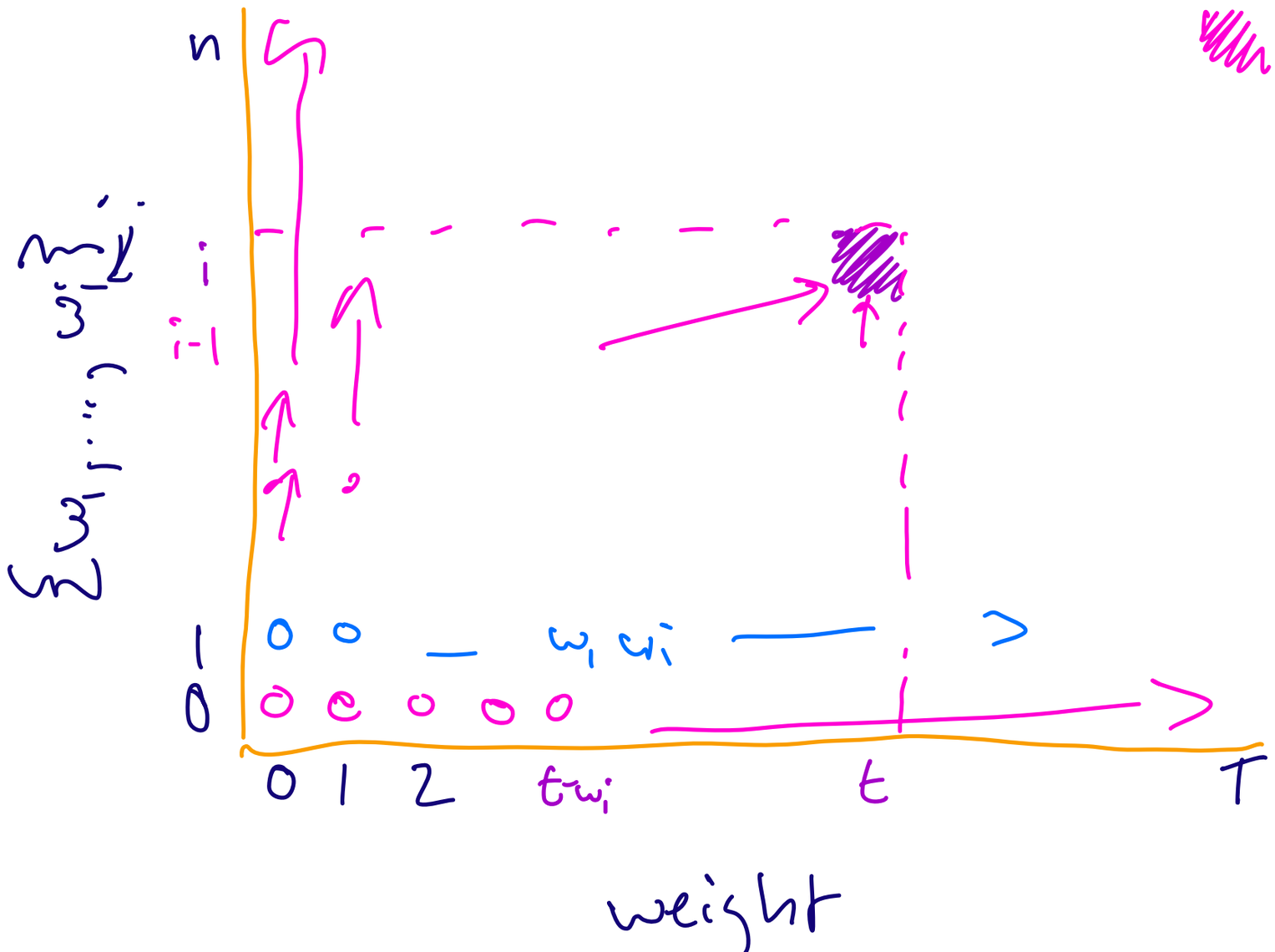
    If $(t - w_i \geq 0)$

      Set $M(i, t) = \max(M[i-1, t], M[i-1, t-w_i] + w_i)$

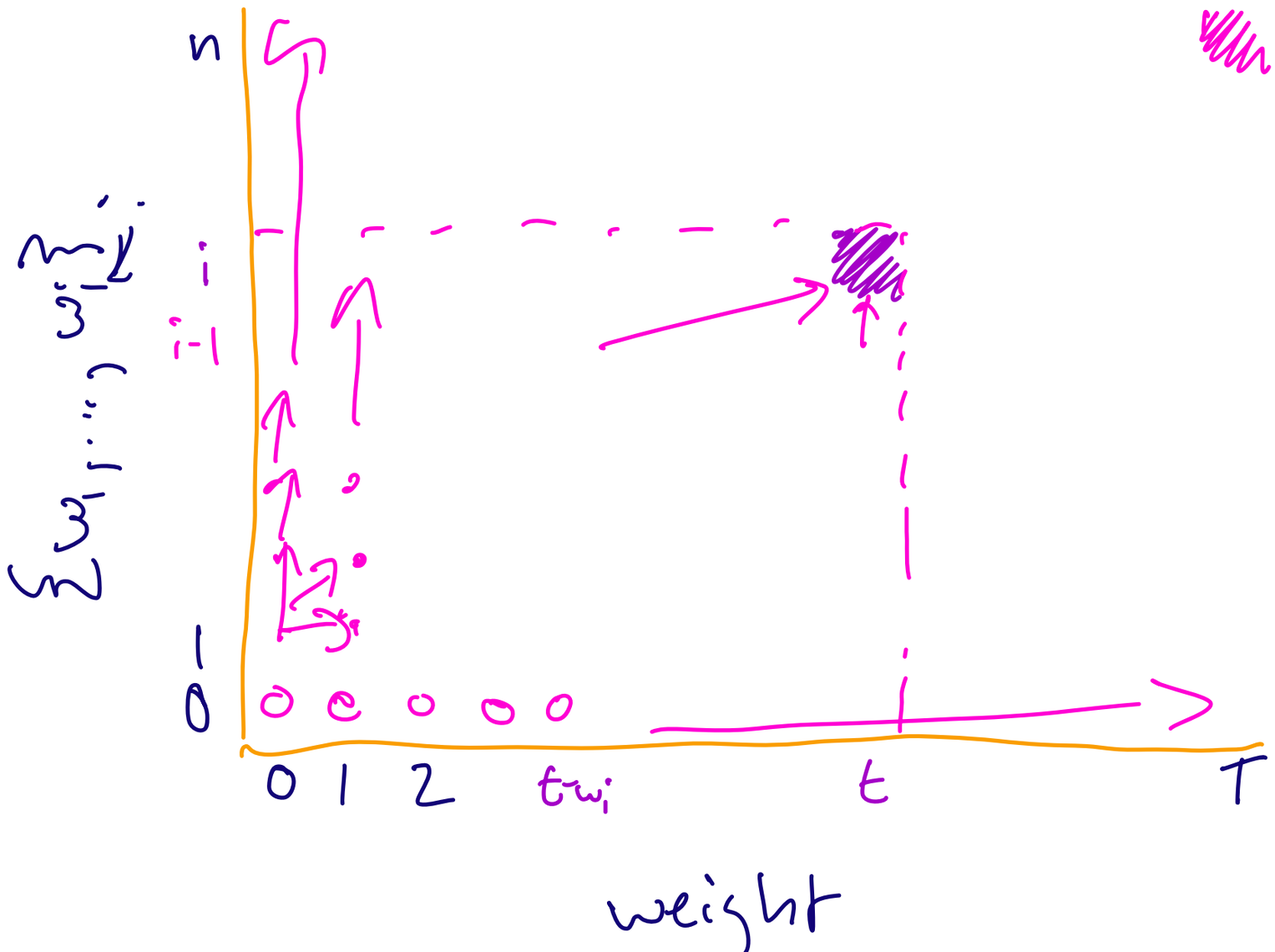
    Else

      Set $M[i, t] = M[i-1, t]$

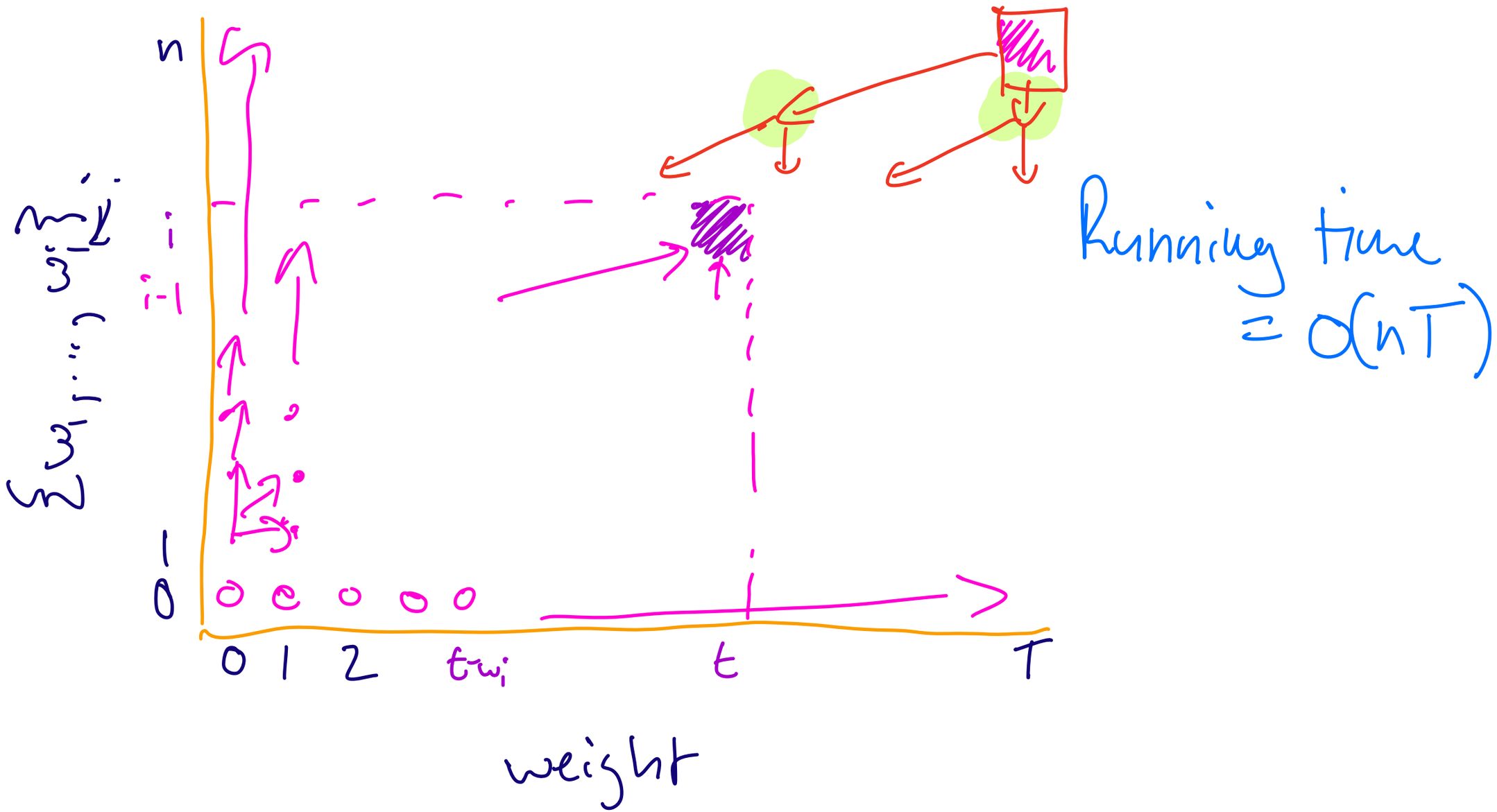
Return $M[n, T]$.



Running time
$= O(nT)$



Running time
$= O(nT)$


n
i
i-1
1
0
0 1 2
t_{w_i}
t
T
weight
Running time
$= O(nT)$

$T = 100 \qquad w_j = \pi$

$$\text{SubsetSum}(j-1, 100)$$

$$\text{SubsetSum}(j-1, 100-\pi)$$

If weights are $\notin \mathbb{Z}$, this doesn't work.

↑ memoization.

Knapsack

Input: $\{(w_i, v_i)\}$ $1 \le i \le n$. Threshold T.

Output: $S \subseteq \{1, \ldots, n\}$ such that

$$\sum_{i \in S} w_i \le T,$$

and $\sum_{i \in S} v_i$ is maximal