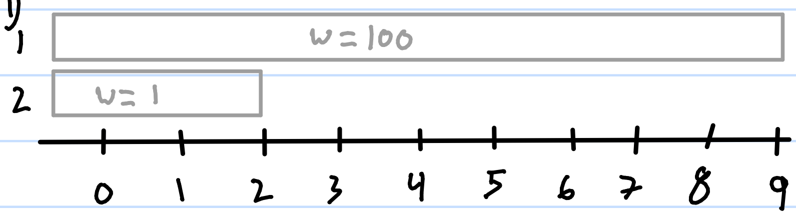
Dynamic Programming

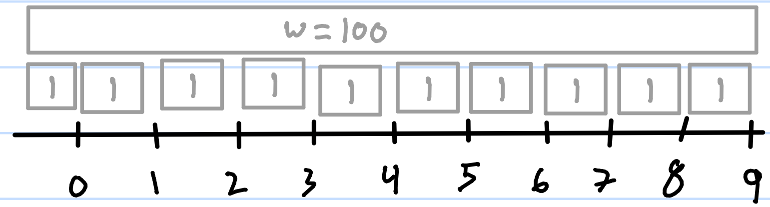
STEVEN.KORDONOWY@COLORADO.EDU

1. Weighted Interval Scheduling
2. Longest Increasing Subsequence
   - Unravelling
3. Edit distance
   - Top-down vs bottom-up

Weighted Interval Scheduling

1)





WEIGHTED INTERVAL SCHEDULING

In: n requests $\{(s_i, f_i, w_i)\}_{i=1}^{n}$ ordered by f_i

Out: Int max weight possible from requests

for $i \in \{1, \ldots, n\}$:

  $p_i \leftarrow \text{argmax}\{j < i : f_j \leq s_i\}$ // argmax $\emptyset = 0$ here

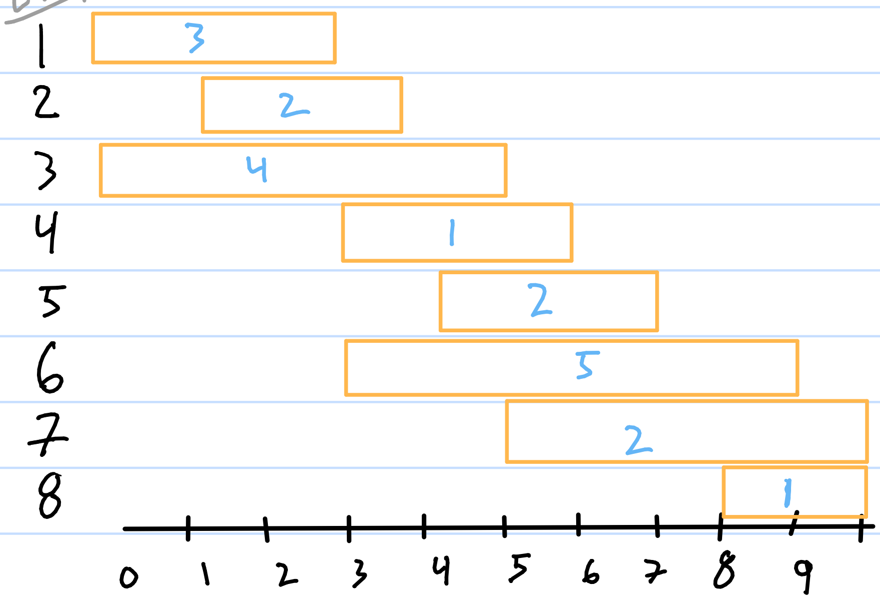
$[\,OPT(0) \leftarrow 0$ // Base Cases

for $j \in \{1, \ldots, n\}$:

  $OPT(j) \leftarrow \max\{\underbrace{w_j + OPT(p_j)}_{\text{use } j}, \underbrace{OPT(j-1)}_{\text{don't use } j}\}$

Return $OPT(n)$

Ex



j	0	1	2	3	4	5	6	7	8
w_j	0	3	2	4	1	2	5	2	1
p_i	0	0	0	0	1	2	1	3	5
$OPT(j)$	0	3	3	4	4	5	8	8	$\boxed{8}$

$OPT(6) = \max\{5 + OPT(1), OPT(5)\}$
$= \max\{8, 5\} = 8$

$OPT(7) = \max(2 + OPT(3), OPT(6))$
$= \max(6, 8) = 8$

$OPT(5) = \max\{2 + OPT(2), OPT(4)\}$
$= \max\{5, 4\} = 5$

Longest Increasing Subsequence

Given some list of integers $(a_1,\dots,a_n)$, what is the longest subsequence $(b_1,\dots,b_m) \subseteq (a_1,\dots,a_n)$ s.t. $b_{i-1} \le b_i \le b_{i+1}$ $\forall i$?

Ex: 5 2 8 6 3 6 9 7

LIS

In: $(a_1,\dots,a_n) \subseteq \mathbb{N}^n$

Out: Int length of LIS # of nodes

$G=(U,V) \longleftarrow$ constructDag$(a_1,\dots,a_n)$ // $U=\{a_1,\dots,a_n\}$, $E=\{a_ia_j : a_i<a_j \text{ \& } i<j\}$

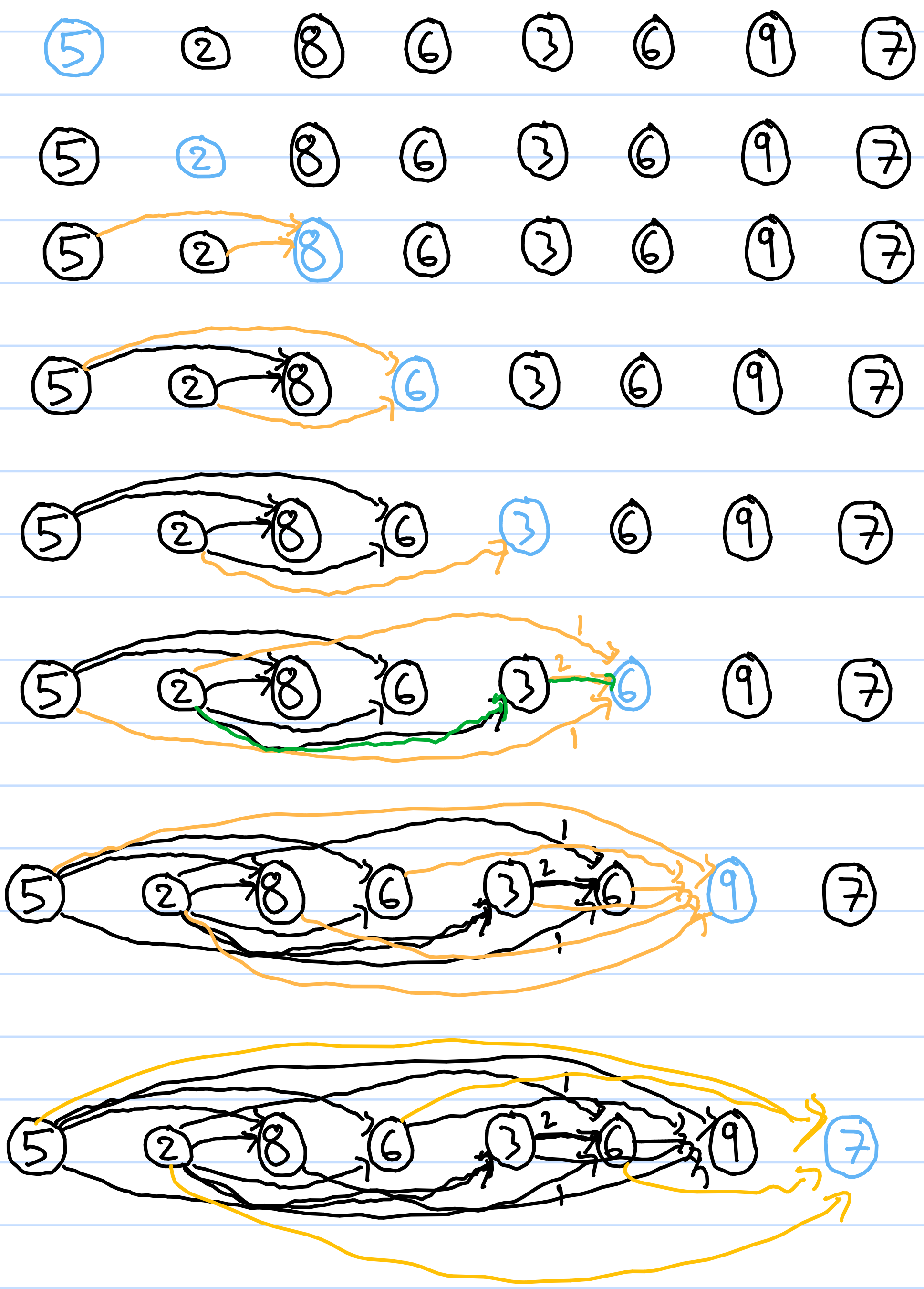
for $v \in U$: // Iterate over vertices

  $L(v) \leftarrow 1 + \max\{L(u) : uv \in V\}$ // Find longest path of nodes u with edges directed at v

Return $\max\{L(i) : i \in V\}$

1 2 3 4 5 6 7 8

colors



$L(1)=1$

$L(2)=1$

$L(3)=1+\max\{L(1),L(2)\}=1+1=2$

$L(4)=2$

$L(5)=2$

$L(6)=1+\max(L(1),L(2),L(5))=1+2=3$

$L(7)=1+L(6)=1+3=4$

$L(8)=1+L(6)=4$

$\max = 4$

Longest Increasing Subsequence What about the items themselves?

LIS

In: $(a_1, \ldots, a_n) \subseteq \mathbb{N}^n$
Out: Int length of LIS

$G = (U, V) \longleftarrow \text{constructDag}(a_1, \ldots, a_n)$
for $v \in U$: // Iterate over vertices
  $L(v) \leftarrow 1 + \max\{L(u) : uv \in V\}$ // Find longest path of nodes u with edges directed at v
Return $\max\{L(i) : i \in V\}$

LIS-Items

In: $(a_1, \ldots, a_n) \subseteq \mathbb{N}^n$
Out: Int length of LIS

$G = (U, V) \longleftarrow \text{constructDag}(a_1, \ldots, a_n)$
for $v \in U$:
  $\text{prev}(v) \longleftarrow \text{null}$ } Init all nodes' previous value to null
for $v \in U$: // Iterate over vertices
  $u \leftarrow \text{argmax}\{L(u) : uv \in V\}$ // Find longest path of nodes u with edges directed at v
  $L(v) \leftarrow 1 + L(u)$
  $\text{prev}(v) \longleftarrow u$
$V_{max} \leftarrow \text{argmax}\{L(i) : i \in V\}$
Return $\text{Unravel}(V_{max})$

Unravel

In: Vertex v
Out: Sequence of nodes from v backwards

$\ell \longleftarrow [v]$
$u \longleftarrow v$
while($\text{prev}(u)$!= null):
  ℓ.append(u)
  $u \longleftarrow \text{prev}(u)$
Return ℓ

Edit Distance

How many edits required to turn SNOWY into SUNNY?

Cost = insertions, deletions, replacements

-	S	N	O	W	-	Y
S	U	N	-	-	N	Y

Cost = 5 edits

S	-	N	O	W	Y
S	U	N	N	-	Y

c = 3

☆ NOT Hamming distance

SNOWY = $x[1,5]$ SUNNY = $y[1,5]$

How does $x[1]$ compare with $y[1]$?

" $x[1,2]$ " $y[1]$?

" $x[1,3]$ " $y[1,4]$?

Given 2 strings $x[1,m]$ & $y[1,n]$

SUBPROBLEM: Compare $x[1,i]$ with $y[1,j]$ as $E(i,j)$

Build up + return $E(m,n)$

How can we make use of smaller subproblems $E(i,j)$ for $i<m, j<n$?

For $i \le m$, $j \le n$, compare rightmost column.

I: $\begin{matrix} x[1] \cdots x[i-1] & x[i] \\ y[1] \cdots y[j] & - \end{matrix}$ Need at least 1 edit + then solve subprob $x[1,i-1]$ w/ $y[1,j]$ $= 1 + E(i-1,j)$

II: $\begin{matrix} x[1] \cdots x[i] & - \\ y[1] \cdots y[j-1] & y[j] \end{matrix}$ 1 edit + $y[1,j-1]$ w/ $x[1,i]$ $= 1 + E(i,j-1)$

III: $\begin{matrix} x[1] \cdots & x[i] \\ y[1] \cdots & y[j] \end{matrix}$ 1 or 0 edits + $x[1,i-1]$ vs $y[1,j-1]$ $= E(i-1,j-1) + \underbrace{\begin{cases} 0 & x[i]=y[j] \\ 1 & \text{o.w.} \end{cases}}_{\delta(i,j)}$

EditDistance

In: Strings $x[1,m]$, $y[1,n]$

Out: Int number of edits to transform x into y

```
for i ∈ {0,...,m}:
    E(i,0) ← i
for j ∈ {0,...,n}:          } Setting up base cases
    E(0,j) ← j

for i ∈ {1,...,m}:
    for j ∈ {1,...,n}:
        E(i,j) ← min{ E(i-1,j)+1
                      E(i,j-1)+1
                      E(i-1,j-1)+ δ(i,j) }     δ(i,j) = { 0  x[i]=y[j]
                                                        { 1  o.w.
Return E(m,n)
```

	-	S	U	N	N	Y
-	0	1	2	3	4	5
S	1	0	1	2	3	4
N	2	1	1	1	2	3
O	3	2	2	2	2	3
W	4	3	3	3	3	3
Y	5	4	4	4	4	3

$E(3,3) = \min(2+1, 1+1, 1+\delta(3,3)) = \min(3,2) = 2$

$E(5,4) = \min(4+1, 3+1, 3+\delta(5,4)) = 4$

$E(5,5) = \min(4+1, 3+1, 3+\delta(5,5)) = 3$

$E(1,1) = \min\{1+E(0,1), 1+E(1,0), \overset{0}{\delta(1,1)} + \overset{0}{E(0,0)}\}$

$= \min\{2,2,0\}$

$E(1,2) = \min(0+1, 2+1, 1+\delta(1,2)) = 1$

$E(1,3) = \min(1+1, 3+1, 2+\delta(1,3)) = \min(2,4,3) = 2$

$E(2,2) = \min(1+1, 1+1, 0+\delta(2,2)) = \min(2,1) = 1$

$E(2,3) = \min(1+1, 2+1, 1+\delta(2,3)) = \min(2,3,1) = 1$