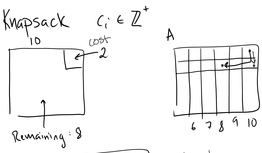
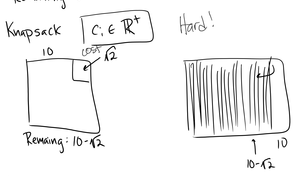
11: DP flexibility

Saturday, April 24, 2021
4:13 PM

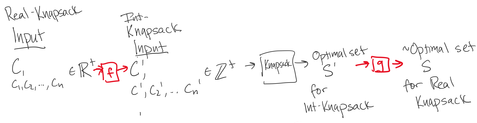




output set should fit in knapsack but might not be optimal value!

Group Work:

- Create alg to approximately solve Knapsack with real costs, capacity by:
  Reducing Knapsack with real costs to Knapsack with integer costs
- Modify reduction to add parameter M, where if M is larger, approximation becomes better.

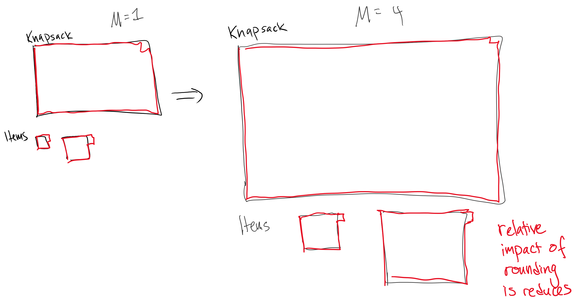


Choose M. Bigger M → closer to optimal
↘ longer runtime

f: $c_i' = \lfloor c_i M \rfloor$
$c_i' = \lceil c_i M \rceil$
Reduction

g: $S = S'$

Result: S will always fit in knapsack, but maybe not optimal
→ Knapsack too small
→ Items too big
} solutions could be valid in original that aren't in reduction



Runtime:

f: $O(n)$
g: $O(i)$
Knapsack: $O(nC') = O(nMC)$ ← Better approximation takes more time!

Why dependence on C is good to know!

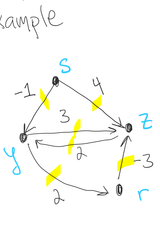
Initialize $n \times n$ array A containing ∞'s.
$A[s,0] = 0$
for $i = 1$ to $n-1$:
  for $v \in V$:
    // Find minimum among options!
    for $u \in W[v]$
      if $A[u, i-1] + w[u,v] < A[v,i]$: $A[v,i] \leftarrow A[u,i-1] + w[u,v]$ } $O(i)$

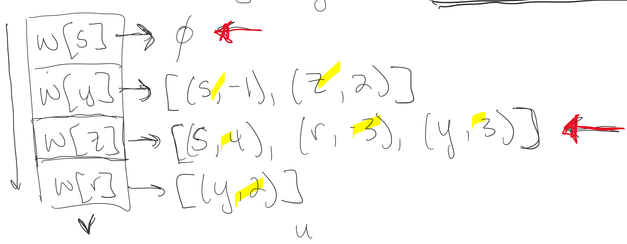
← $O(n)$

Over these two loops, iterate $O(m)$ once per edge
↑ # of edges in graph

Example

W = reverse adjacency list (input to algorithm)





(Amortized analysis)

Runtime: $O(nm)$ $m = O(n^2)$

"Sparse Graph"