Approximation Algorithms

So far, we have three examples of NP-complete problems with $\frac{1}{2}$-approximations.

- NODE COVER ($\frac{1}{2}$-approx)
- MAXSAT ($\frac{1}{2}$-approx or $\frac{1}{2^k}$-approx)
- MAX-CUT ($\frac{1}{2}$-approx)

No all problems are so well-behaved

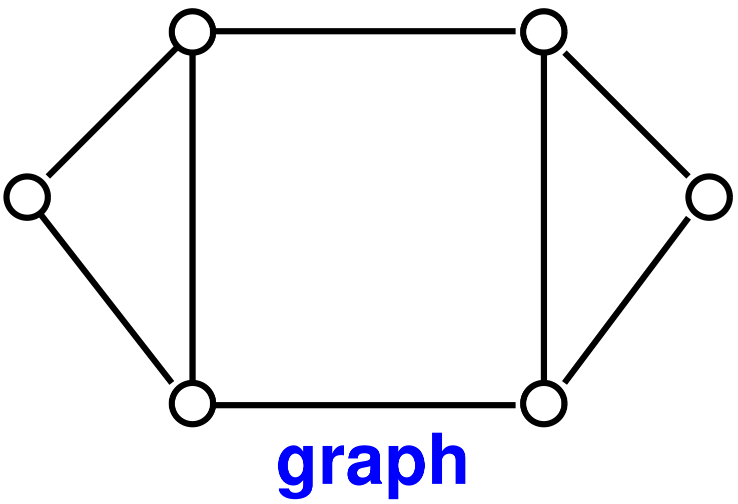
TSP

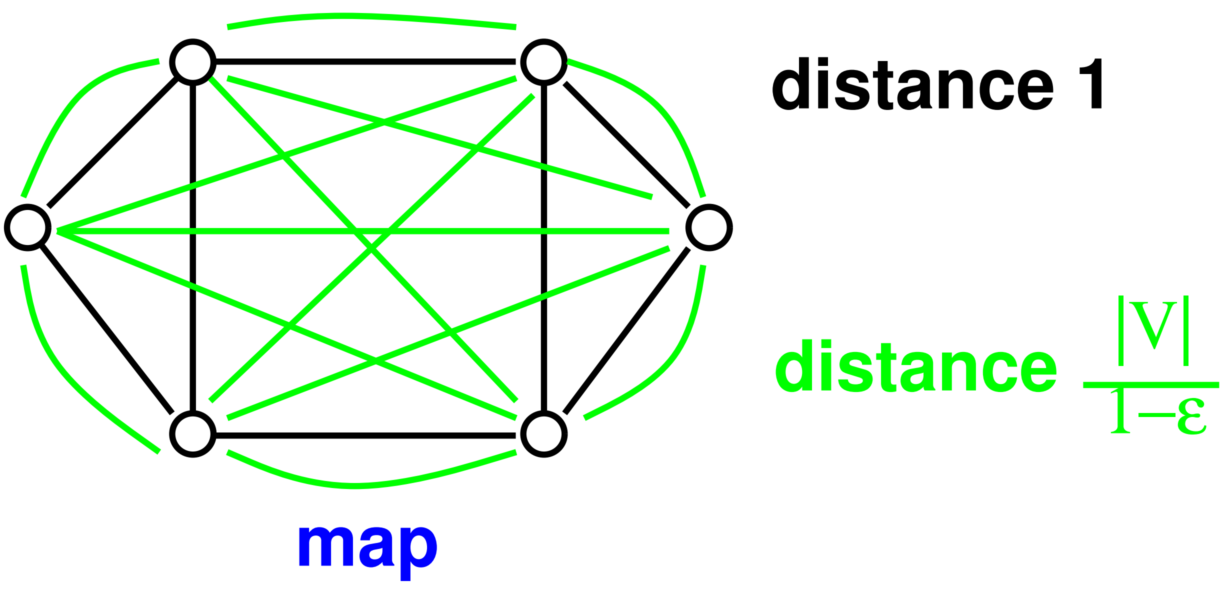
Claim: TSP has no polynomial-time ϵ-approximation algorithm for any $\epsilon < 1$. (Except if $P =$ NP.)

Strategy:

- assume otherwise, and
- derive polynomial-time algorithm for HAMCIRCUIT
- which is, of course, NP-complete.

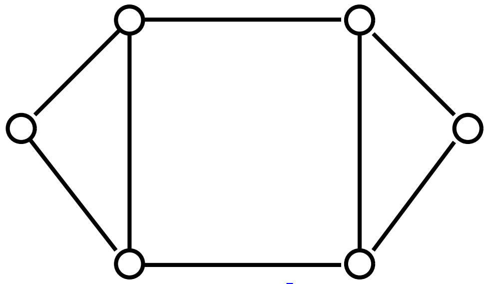
Construction



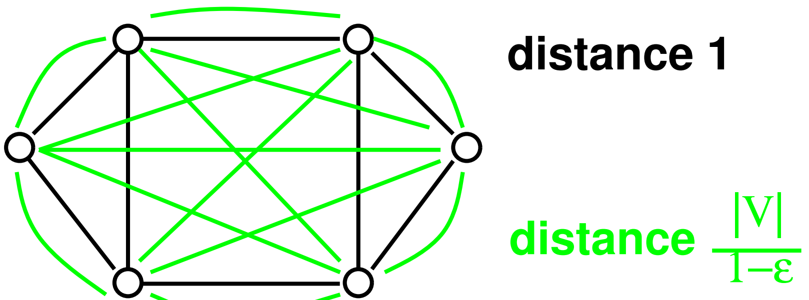


Transform any graph into a map.

Construction



graph



map

Ask ϵ-approximation algorithm for traveling salesman tour.

- If path length is $|V|$, then
  - all edges traversed have distance 1, and
  - form a Hamiltonian circuit in graph
- If path length $> |V|$, then
  - all some edge has distance $\frac{|V|}{1-\epsilon}$
  - total distance $\geq \frac{|V|}{1-\epsilon}$
  - optimal distance $\leq 1-\epsilon$ times path distance
  - optimal distance $> |V|$
  - no Hamiltonian circuit exists.

Special Cases

Suppose all distances are either 1 or 2.

- we proved this restricted problem NP-complete.

Claim: The (1,2)-TSP problem has an ϵ-approximation

Proof: All tours have length at most twice optimal!

Remark: $\frac{1}{7}$-approximation known.

Knapsack

We are given a sequence of items, $x_1, \ldots, x_n$, each has

- an integer *weight* w_i and
- an integer *value* v_i.

We are also given an integer *maximum weight* $W \geq w_i$,

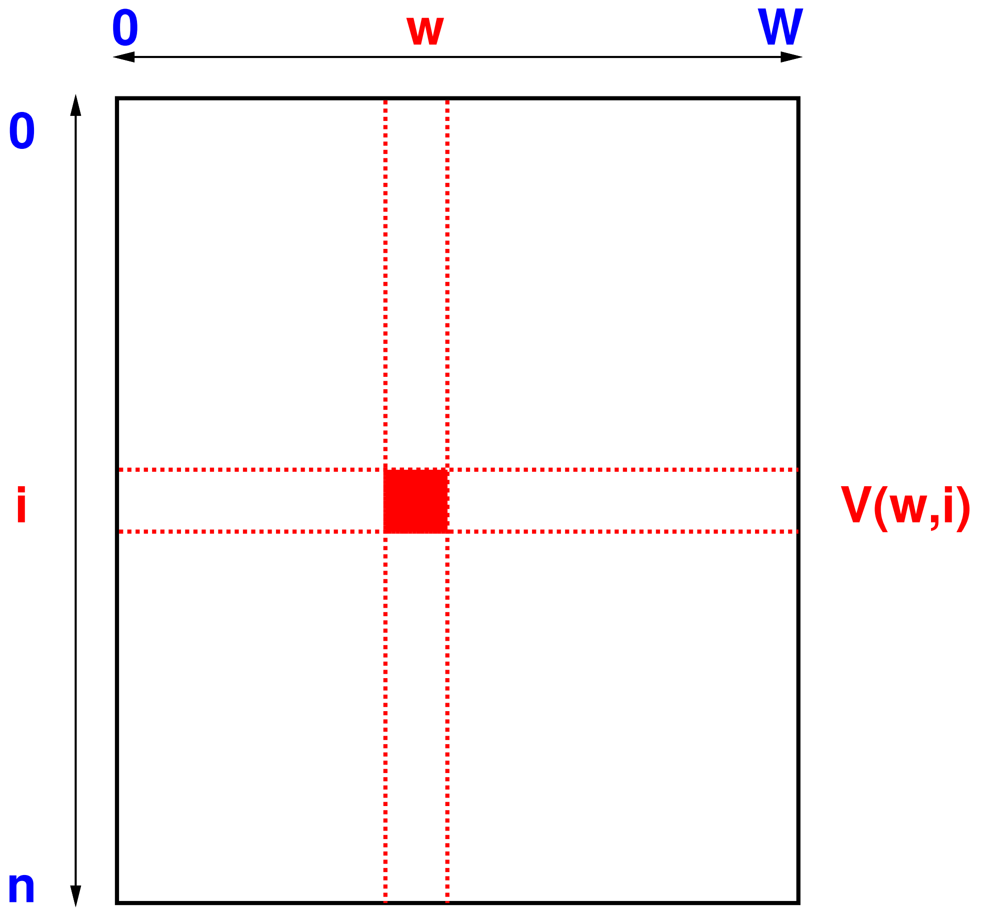
Problem: find a subset S of items such that

- total weight $\leq W$
- total value maximized

Related to SUBSET-SUM.

This problem is NP-complete.

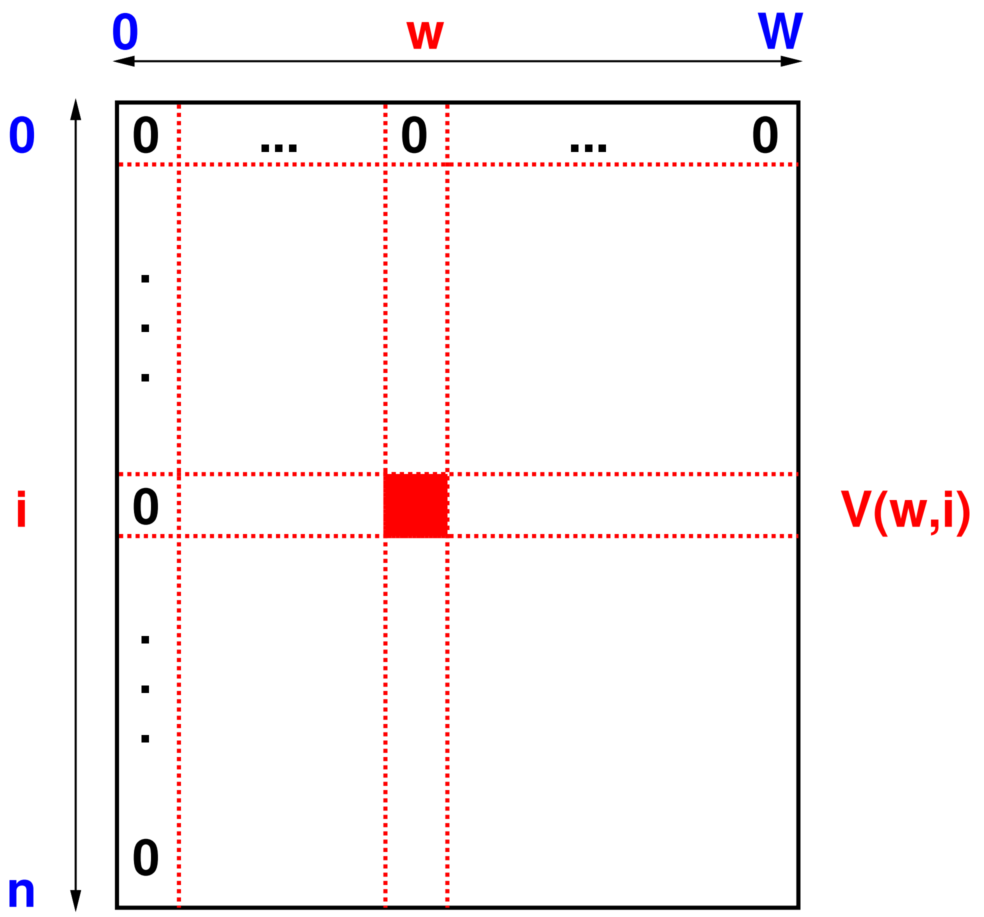
An $O(nW)$ KNAPSACK Algorithm



Try a little dynamic programming . . .

- let $V(w,i)$ be the largest value attainable from
- the first i items, so that their
- weight does not exceed w.

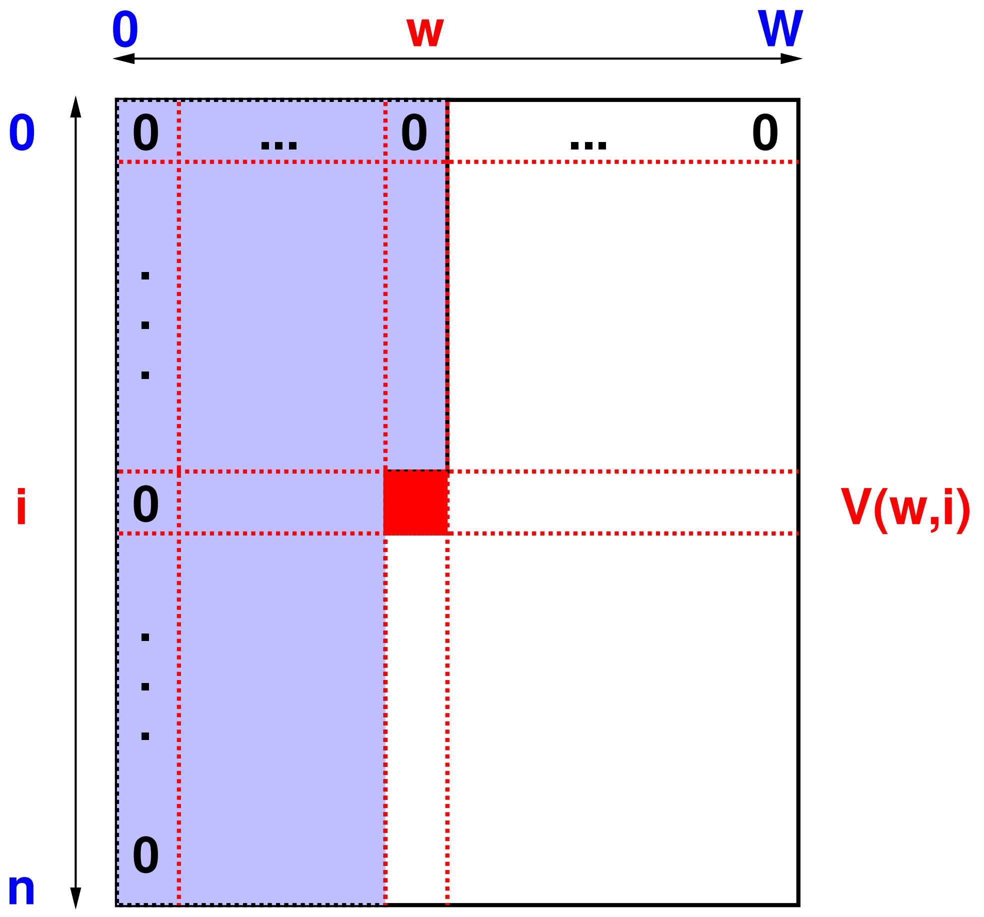
An $O(nW)$ KNAPSACK Algorithm



Initially,

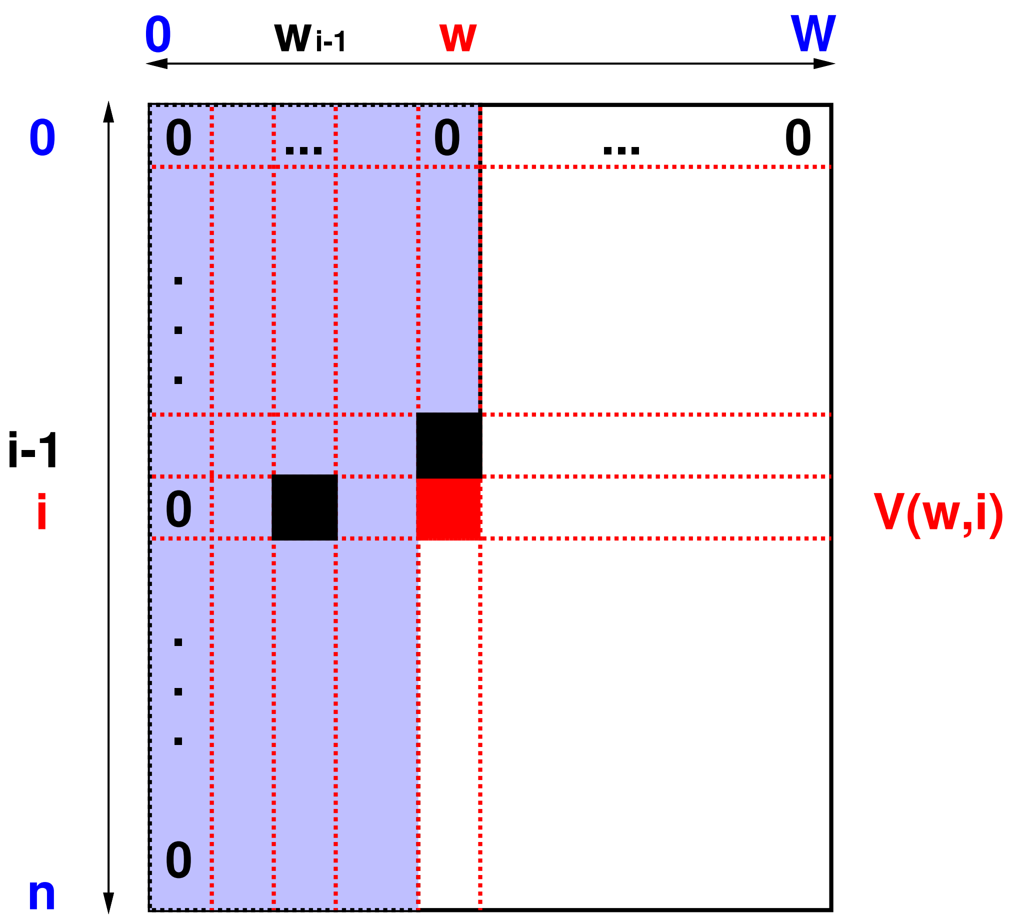
- $V(0, i) = 0$, and
- $V(w, 0) = 0$.

An $O(nW)$ KNAPSACK Algorithm



We fill in $V(w,i)$ like this.

An $O(nW)$ KNAPSACK Algorithm



$$V(w, i+1) = \max(V(w,i), v_{i+1} + V(w - w_{i+1}, i))$$

- constant number of operations per entry, and
- nW entries in all.
- Solution is greatest entry in table.

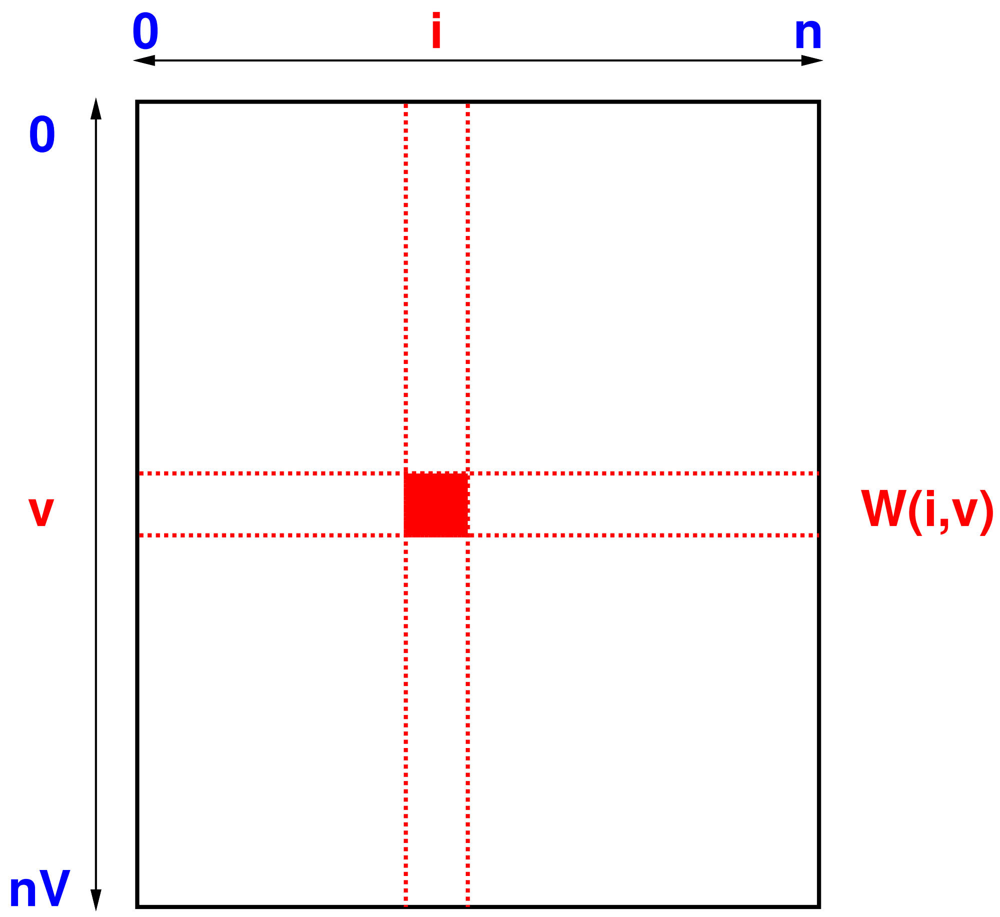
An $O(nW)$ KNAPSACK Algorithm

Claim: This $O(nW)$ KNAPSACK algorithm is polynomial . . . *knot!*

Input length is $O(n \log W)$, not $O(W)$.

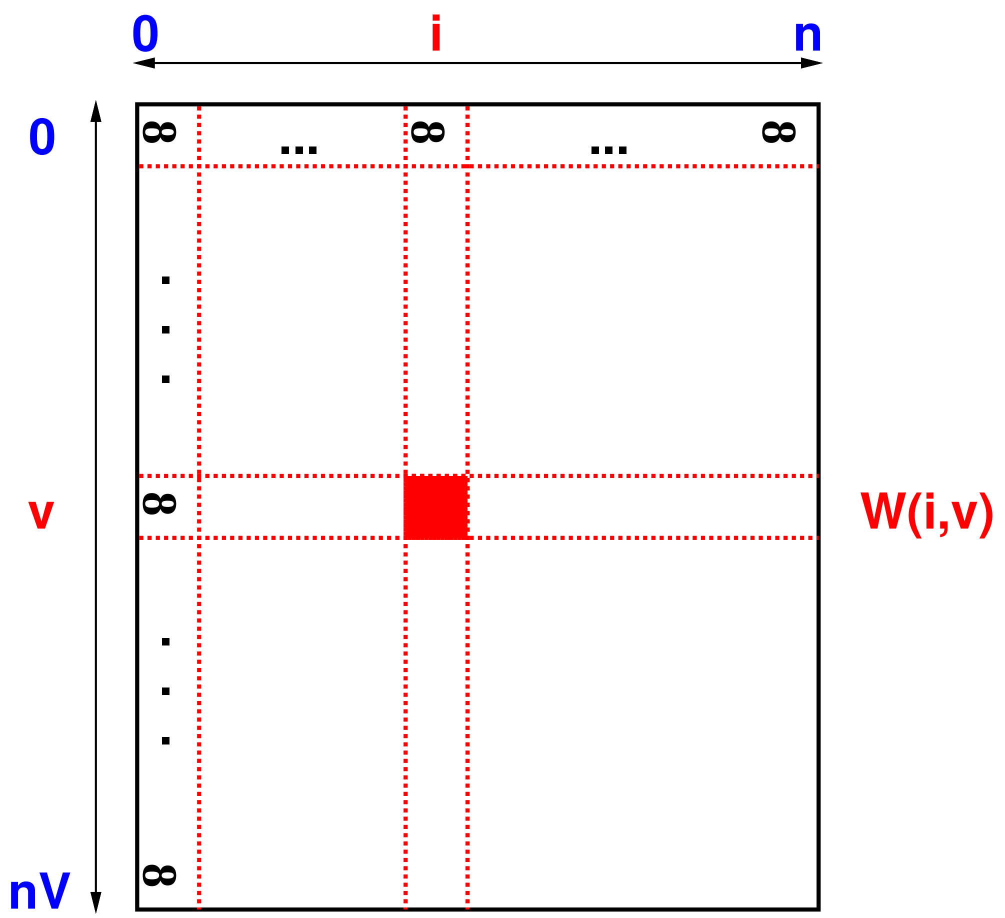
Another Pseudo-polynomial KNAPSACK Algorithm

Let $V = max(v_1, \ldots, v_n)$.



- let $W(i, v)$ be the minimum weight attainable from
- the first i items, so that their
- value is exactly v.

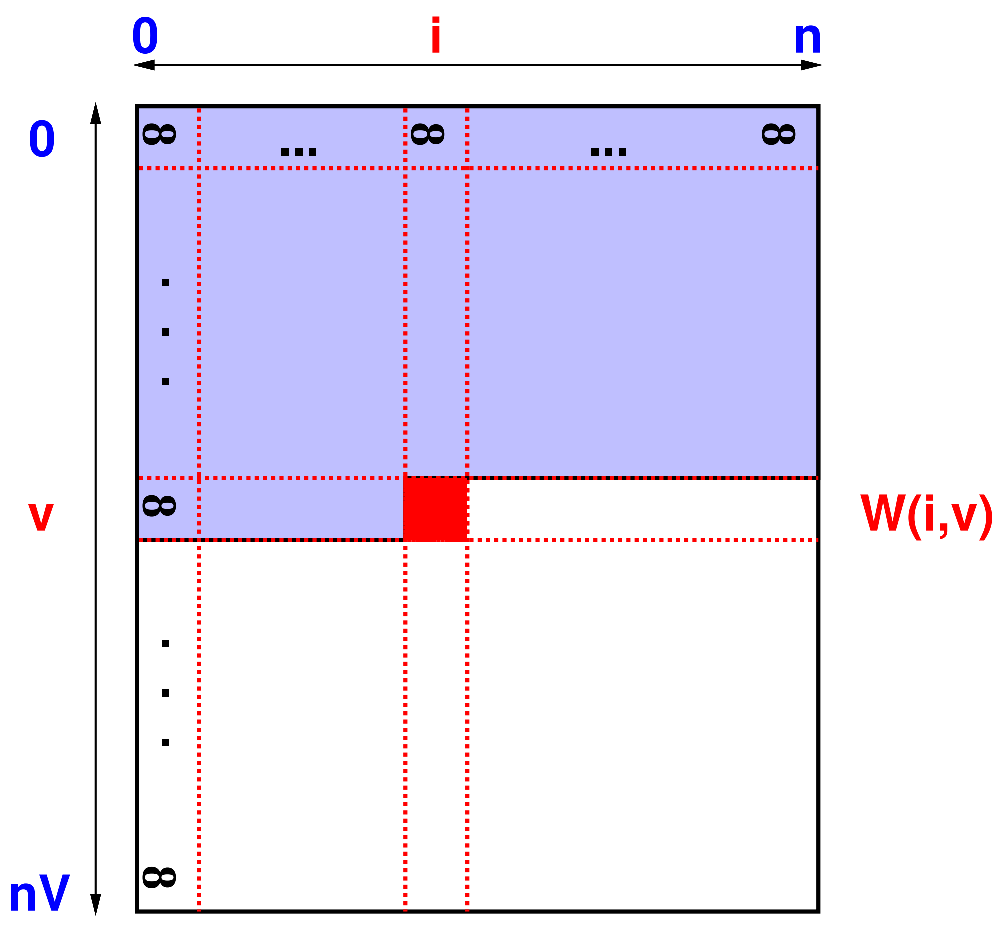
Another Pseudo-polynomial KNAPSACK Algorithm



Initially,

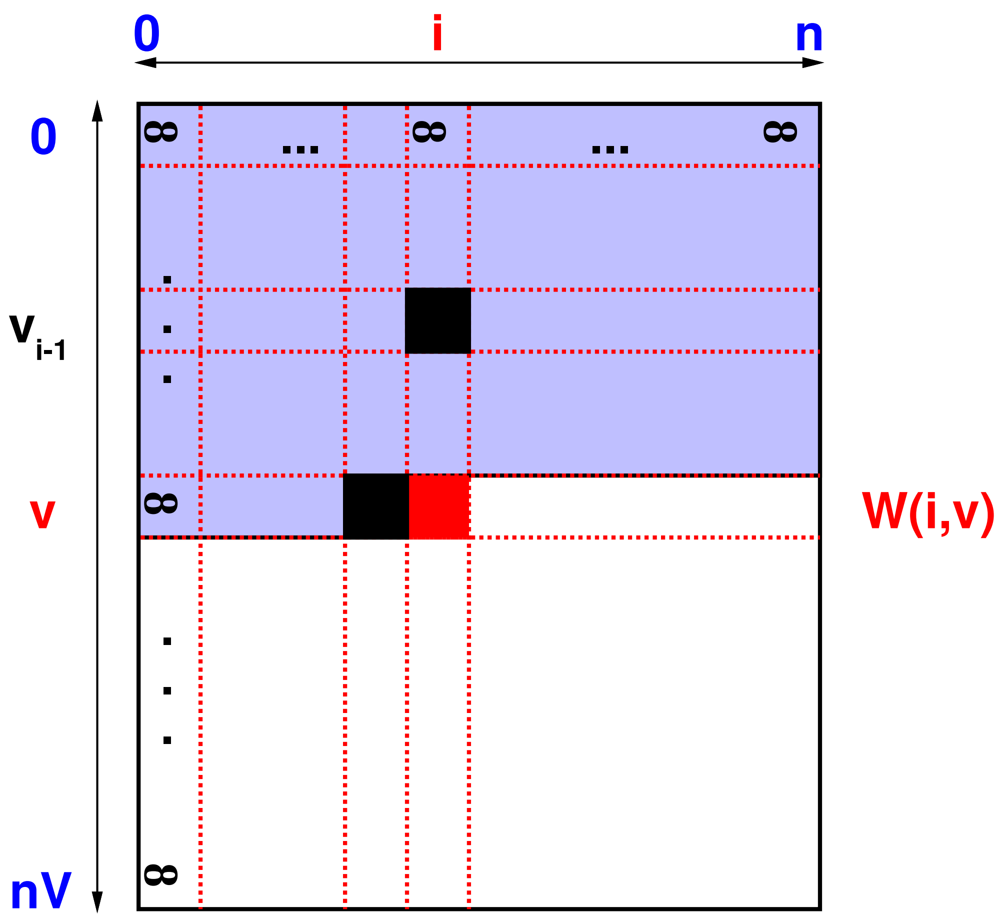
- $W(i, 0) = \infty$, and
- $W(0, v) = \infty$.

Another Pseudo-polynomial KNAPSACK Algorithm



We fill in $W(i, v)$ like this.

Another Pseudo-polynomial KNAPSACK Algorithm



$$W(i+1, v) = \min(W(i, v), w_{i+1} + W(i, v - v_{i+1}))$$

- constant number of operations per entry, and
- n^2V entries in all.
- solution is largest v such that $W(i, v) \leq W$.

Also pseudo-polynomial.

Basic Idea

This algorithm is pseudo-polynomial, because

- input has size $O(\log V)$, but
- running time is V, which is
- exponentially larger.

What if we take approximations to the v_i?

- given instance $x = (w_1, \ldots, w_n, W, v_1, \ldots, v_n)$
- derive $x' = (w_1, \ldots, w_n, W, v'_1, \ldots, v'_n)$
- by replacing b least significant digits of each v_i with 0s

$$v'_i = 2^b \left\lfloor \frac{v_i}{2^b} \right\rfloor$$

We will choose b later . . .

More Basic Idea

Given instance

$$x = (w_1, \ldots, w_n, W, v_1, \ldots, v_n),$$

with optimal solution S,

derive

$$x' = (w_1, \ldots, w_n, W, v'_1, \ldots, v'_n).$$

with optimal solution S,

Solution	problem	running time
S	x	$O(n^2V)$
S'	x'	$O\left(\frac{n^2V}{2^b}\right)$

Some Inequalities

$$\sum_{i \in S} v_k \geq \sum_{i \in S'} v_k \geq \sum_{i \in S'} v_i'.$$

Because

- S is an optimal solution for x.
- $v_i' \leq v_i$.

Some Inequalities

We have seen that:

$$\sum_{i \in S} v_i \geq \sum_{i \in S'} v_i \geq \sum_{i \in S'} v'_i.$$

We now claim that:

$$\sum_{i \in S'} v'_i \geq \sum_{i \in S} v'_i.$$

Because S' is an optimal solution for x'.

Some Inequalities

We have seen that:

$$\sum_{i \in S} v_i \geq \sum_{i \in S'} v_i \geq \sum_{i \in S'} v'_i \geq \sum_{i \in S} v'_i$$

We now claim that:

$$\sum_{i \in S} v'_i \geq \sum_{i \in S} (v_i - 2^b).$$

Because $v'_i \geq v_i - 2^b$.

Some Inequalities

We have seen that:

$$\sum_{i\in S} v_i \geq \sum_{i\in S'} v_i \geq \sum_{i\in S'} v_i' \geq \sum_{i\in S} v_i' \geq \sum_{i\in S} (v_i - 2^b)$$

We now claim that:

$$\sum_{i\in S} (v_i - 2^b) \geq \sum_{i\in S} v_i - n2^b.$$

Because $|S| \leq n$.

Integrating

In short,

$$\sum_{i \in S'} v_i \geq \sum_{i \in S} v_i - n2^b.$$

Our approximate solution is off by $\leq n2^b$.

Integrating

In short,

$$\frac{\sum_{i \in S'} v_i}{\sum_{i \in S} v_i} \geq \frac{\sum_{i \in S} v_i - n2^b}{\sum_{i \in S} v_i} = 1 - \frac{n2^b}{\sum_{i \in S} v_i}$$

V is a lower bound on $\sum_{i \in S} v_i$,

$$\frac{n2^b}{\sum_{i \in S} v_i} \geq \frac{n2^b}{V} Andso$$

$$\frac{\sum_{i \in S'} v_i}{\sum_{i \in S} v_i} \geq 1 - \frac{n2^b}{V}$$

It follows that $\epsilon = \frac{n2^b}{V}$.

Good News

User provides desired ϵ.

- we provide ϵ-algorithm that
- truncates last $b = \left\lceil \log \frac{\epsilon V}{n} \right\rceil$ bits
- takes time $O(\frac{n^2 V}{2^b}) = O(\frac{n^3}{\epsilon})$.

Claim: For any $\epsilon > 0$ there is a polynomial-time ϵ-approximation algorithm for KNAPSACK.

Schemes

Definition: A *polynomial-time approximation scheme* for an optimization problem is

- an algorithm A
- for any $\epsilon > 0$, and
- any problem instance x
- returns a polynomial-time algorithm
  - polynomial in $|x|$
  - but depending on $\frac{1}{\epsilon}$.

If the polynomial (in $|x|$) depends *polynomially* on $\frac{1}{\epsilon}$, then the scheme is *fully polynomial*.