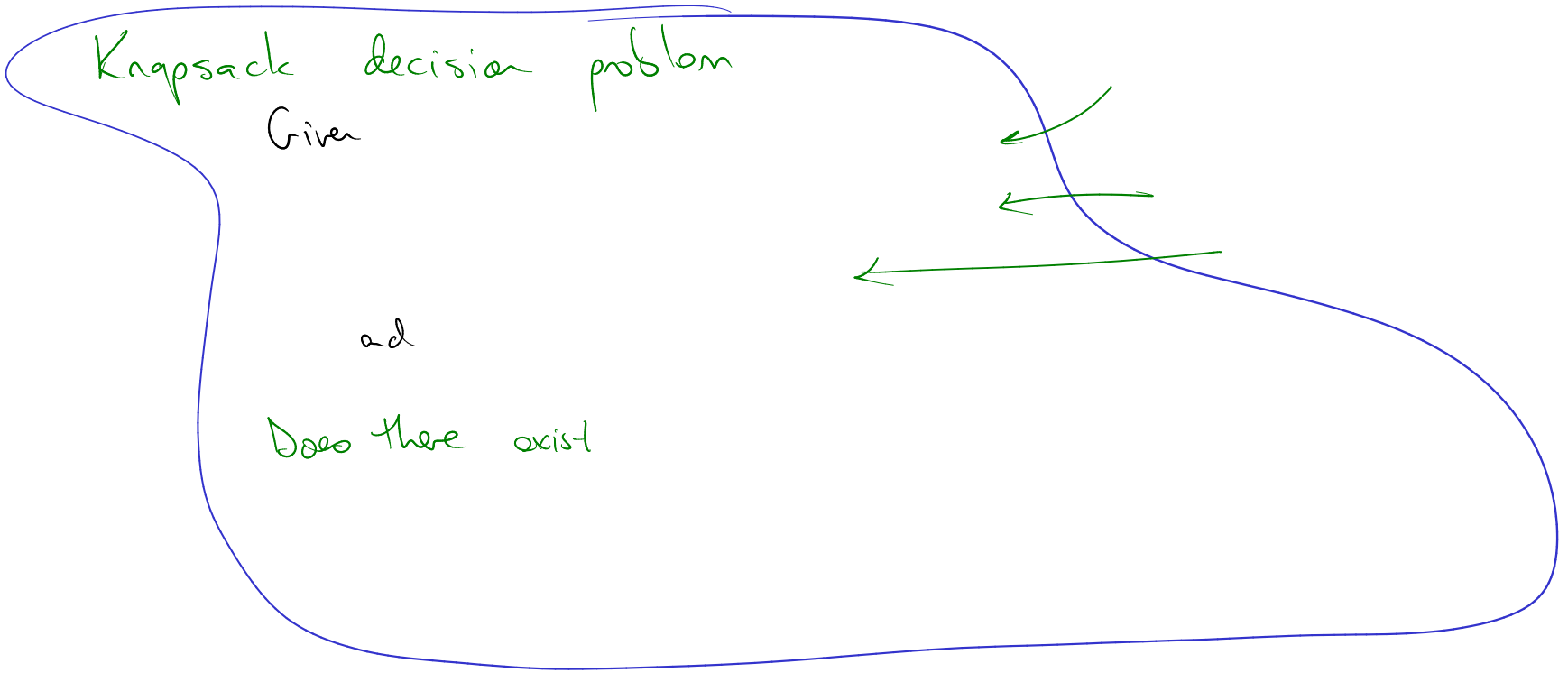
Math 343, lecture 18

① The Knapsack problem

Suppose you have

Here are some problems you might want to solve



You might also want

Knapsack search problem

Given profits $p_0, p_1, \ldots, p_{n-1}$
weights $w_0, w_1, \ldots, w_{n-1}$
max capacity M
and target profit P

Find

But why choose a target profit in advance – lets find the optimal profit

Knapsack optimal value problem

Given profits $p_0, p_1, \ldots, p_{n-1}$

weights $w_0, w_1, \ldots, w_{n-1}$

and max capacity M

Find

subject to

Again you might actually want the n-tuple which gives the max

Knapsack optimization problem

Given profits $p_0, p_1, \ldots, p_{n-1}$

weights $w_0, w_1, \ldots, w_{n-1}$

and max capacity M

Find

maximizing

and subject to

This is a prototypical optimization problem

Optimization vocabulary

constraint

feasible solution

objective function

(2) Backtracking algorithms

But how do we solve the Knapsack problem?
Lets focus on the optimization problem.

The most naive thing we could do is

How do we want to do that

The one we want in order to lead into what comes next is to traverse the generating tree

Generating tree for binary strings

ε

A depth first traversal of this tree will give

The tree also makes a recursive implementation very natural

Optimized vocabulary

state space tree

state space (for the problem of size n)

Algorithm Naive Knapsack

global n, $(w_0, \ldots, w_{n-1})$, $(p_0, \ldots, p_{n-1})$ input to the overall problem

$x = (x_0, \ldots, x_\ell)$ current feasible solution

optx best x so far

optP best profit so far

input ℓ current length

if $\ell = n$

else

This is not so good

How do we do better?

Call this pruning the state space tree

Algorithm Pruned Knapsack

global $n, (w_0, \ldots, w_{n-1}), (p_0, \ldots, p_{n-1})$

$x = (x_0, \ldots, x_l)$

opt x

opt P

input l, curW

if $l = n$

else

Every backtracking algorithm has this same basic shape

```
Metaalgorithm Backtrack
    global   x = (x_0, x_1, ...)
    input l
    if (x_0, ..., x_{l-1}) is feasible
        then process it
    Compute C_l
    for x_l ∈ C_l
        Backtrack(l+1)
```

Where P_l is

and $C_l \subseteq P_l$ is

possibility set

choice set

(3) Next time

Backtracking for maximal cliques