# Algorithmic Problems 2

Andreas Klappenecker
[based on a lecture by Avrim Blum]

# Motivation

Suppose that you have a homework assignment consisting of seven parts A, B, ..., G. Each part has a certain value of points and takes a certain time to complete. For example,

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| value | 7 | 9 | 5 | 12 | 14 | 6 | 12 |
| time | 3 | 4 | 2 | 6 | 7 | 3 | 5 |

If you have 15 hours, which parts would you do?

# Knapsack

In the knapsack problem, we are given a set of n items, where each item i is specified by a size $s_i$ and a value $v_i$. You are also given a upper bound S on the total of the sizes (namely, the size of the knapsack).

Goal: Find a subset of the items of maximum total value such that the sum of their sizes is at most S.

# Problem

Find an (efficient) algorithm to solve the knapsack problem.

[Hint: Write a recursive procedure Value(n, S) that will select the maximum value among the n items. Assume that the values are stored in an array v[1..n] and the sizes in a array s[1..n].]

# Hint

Either include the last element or don't.

# Recursive Algorithm

```
// Recursive algorithm: either we use the last element or we don't.
Value(n,S)    // S = space left, n = # items still to choose from
{
  if (n == 0) return 0;
  if (s_n > S) result = Value(n-1,S); // can't use nth item
  else result = max{v_n + Value(n-1, S-s_n), Value(n-1, S)};
  return result;
}
```

We need exponential time, since at each iteration, we have two recursive calls in the worst (but normal) case.

There are at most $O(nS)$ values!

Now speed up the recursive algorithm! Which algorithm design method can  you use?

# Dynamic Programming (Memoization)

```
Value(n,S)
{
  if (n == 0) return 0;
  if (arr[n][S] != unknown) return arr[n][S];  // <- added this
  if (s_n > S) result = Value(n-1,S);
  else result = max{v_n + Value(n-1, S-s_n), Value(n-1, S)};
  arr[n][S] = result;                              // <- and this
  return result;
}
```

# Are We Done Yet?

How can you get the actual items that led to the solution?

The knapsack decision problem (can we find items with value of value >= v without exceeding the size S?) is NP complete.

Is this a contradiction?