Algorithms & Complexity

Space Complexity. Deterministic Space

Anton Bryl - abryl@computing.dcu.ie

CA313@Dublin City University. 2009-2010.

November 19, 2009

Space complexity

Two main characteristics for programs

- Time complexity: $\simeq$ CPU usage
- Space complexity: $\simeq$ RAM usage

Space complexity: an informal definition

Definition (Space complexity)

The *space complexity* of a program (for a given input) is the number of elementary objects that this program needs to store during its execution.
This number is computed with respect to the size n of the input data.

Space complexity: a formal definition

Definition (Space complexity)

For an algorithm T and an input x, $DSPACE(T, x)$ denotes the number of cells used during the (deterministic) computation $T(x)$. We will say that $DSPACE(T) = O(f(n))$ if $DSPACE(T, x) = O(f(|x|))$ where $|x|$ is the length of x.

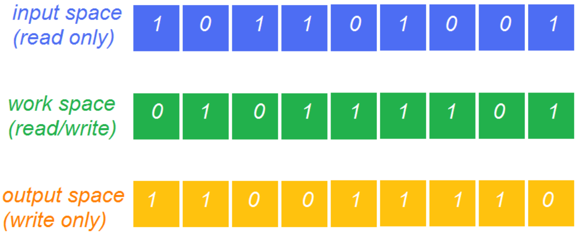
Space complexity: a formal definition

Definition (Space complexity)

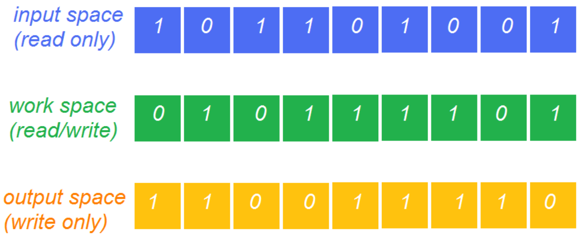
For an algorithm T and an input x, $DSPACE(T, x)$ denotes the number of cells used during the (deterministic) computation $T(x)$. We will say that $DSPACE(T) = O(f(n))$ if $DSPACE(T, x) = O(f(|x|))$ where $|x|$ is the length of x.

- $DSPACE(T)$ is undefined whenever $T(x)$ does not halt.

Space complexity: what counts?



Space complexity: what counts?



- When talking about space complexity, we usually talk only about **work** space.

Space complexity: storing an arbitrary number

- How much space does it take to store n?

Space complexity: storing an arbitrary number

- How much space does it take to store n?
- the tape alphabet Γ is a finite set, so a finite number M of tape cells can take at most $|\Gamma|^M$ different values, we cannot store *an arbitrary number* there.

Space complexity: storing an arbitrary number

- How much space does it take to store n?
- the tape alphabet Γ is a finite set, so a finite number M of tape cells can take at most $|\Gamma|^M$ different values, we cannot store *an arbitrary number* there.
- $O(log(n))$ will be enough to store n.

Space complexity: an example

- Let x and y be two vectors of equal length n, $\forall i : x_i \in \{0, 1\}, y_i \in \{0, 1\}$
- Calculate the product:

$$\sum_{i=1}^{n} x_i \times y_i$$

Space complexity: example (continued)

```
sum = 0;
for (i = 0; i < n; i++)
  p[i] = x[i] * y [i];
for (i = 0; i < n; i++)
  sum += p[i];
return sum;
```

Space complexity: example (continued)

```
sum = 0;
for (i = 0; i < n; i++)
  p[i] = x[i] * y [i];
for (i = 0; i < n; i++)
  sum += p[i];
return sum;
```

If considering only work space:

$$T(findMin, n) = size(p) + size(i) + size(sum) = n + 2 \times ln(n) = O(n)$$

Space complexity: example (continued)

```
sum = 0
for (i = 0; i < n; i++)
  sum += x[i] * y [i];
return sum;
```

Space complexity: example (continued)

```
sum = 0
for (i = 0; i < n; i++)
  sum += x[i] * y [i];
return sum;
```

$$T(findMin, n) = size(i) + size(sum) = 2 \times ln(n) = O(ln(n))$$

Logarithmic space complexity

The class L is defined as:

$$L = DSPACE(log(n)).$$

L is the (complexity) class of decision problems that can be solved using a deterministic Turing Machine and a logarithmic amount of space.

Log-space reduction

Definition (Log-space reduction)

Problem A is log-space reducible to problem B, if A can be reduced to B using a log-space Turing machine M.

Polynomial space complexity

The class *PSPACE* is defined as:

$$PSPACE = \cup_{k \in \mathbb{N}} DSPACE(n^k).$$

PSPACE is the (complexity) class of decision problems that can be solved using a deterministic Turing Machine and a polynomial amount of space.

PSPACE-complete problems

- The "hardest" problems in *PSPACE* constitute the *PSPACE*-complete class.
- A problem $A \in PSPACE$ is *PSPACE*-complete if all problems in *PSPACE* are polynomial-time reducible to A.
- This class includes problems from automata theory, computational linguistics (parsing), and game theory.

PSPACE-complete: Canadian traveller problem

- A generalization of the shortest path problem...

PSPACE-complete: Canadian traveller problem

- A generalization of the shortest path problem...
- ...to graphs that are *partially observable*.
- The structure of the graph is gradually revealed while exploring.

Relation between *DTIME* and *DSPACE*

- What can you say about relation between *DTIME* and *DSPACE*?

Relation between *DTIME* and *DSPACE*

- What can you say about relation between *DTIME* and *DSPACE*?
- It is impossible to use more cells than steps in the computation.
- So $DTIME(f(n)) \subseteq DSPACE(f(n))$.

Questions

- Can you name a problem, which is in *PSPACE* but not necessarily in *P*?

Questions

- Can you name a problem, which is in *PSPACE* but not necessarily in *P*?
- Can you prove that $NP \subseteq PSPACE$?