Introduction to Complexity Classes

Marcin Sydow

Definition

TIME(f(n)) TIME(f(n)) denotes the set of languages decided by deterministic TM of TIME complexity f(n)

Definition

SPACE(f(n)) denotes the set of languages decided by deterministic TM of SPACE complexity f(n)

Definition

NTIME(f(n)) denotes the set of languages decided by non-deterministic TM of TIME complexity f(n)

Definition

NSPACE(f(n)) denotes the set of languages decided by non-deterministic TM of SPACE complexity f(n)

Linear Speedup Theorem

Theorem

If L is recognised by machine M in time complexity f(n) then it can be recognised by a machine M' in time complexity $f'(n) = \epsilon f(n) + (1+\epsilon)n$, where $\epsilon > 0$.

Blum's theorem

There exists a language for which there is no fastest algorithm! (Blum - a Turing Award laureate, 1995)

Theorem

There exists a language L such that if it is accepted by TM of time complexity f(n) then it is also accepted by some TM in time complexity $\log(f(n))$.

Basic complexity classes

(the functions are asymptotic)

- $P = \bigcup_{j>0} TIME(n^j)$, the class of languages decided in deterministic polynomial time
- $NP = \bigcup_{j>0} NTIME(n^j)$, the class of languages decided in non-deterministic polynomial time
- $EXP = \bigcup_{j>0} TIME(2^{n^j})$, the class of languages decided in deterministic exponential time
- $NEXP = \bigcup_{j>0} NTIME(2^{n^j})$, the class of languages decided in non-deterministic exponential time

Space complexity classes

- $L = SPACE(log n)$, the class of languages decided in deterministic logarithic space
- $NL = NSPACE(log n)$, the class of languages decided in non-deterministic logarithic space
- $PSPACE = \bigcup_{j>0} SPACE(n^j)$, the class of languages decided in deterministic polynomial space
- $NPSPACE = \bigcup_{j>0} NSPACE(n^j)$, the class of languages decided in non-deterministic polynomial space
- $EXPSPACE = \bigcup_{j>0} SPACE(2^{n^j})$, the class of languages decided in deterministic exponential space
- $NEXPSPACE = \bigcup_{j>0} NSPACE(2^{n^j})$, the class of languages decided in non-deterministic exponential space

Basic relations among complexity classes

(Connections between time, space and non-determinism)

- $\mathrm{TIME}(f(n)) \subseteq \mathrm{NTIME}(f(n))$

Basic relations among complexity classes

(Connections between time, space and non-determinism)

- TIME($f(n)$) $\subseteq$ NTIME($f(n)$), since each deterministic machine is also a non-deterministic one (by definition)
- SPACE($f(n)$) $\subseteq$ NSPACE($f(n)$)

Basic relations among complexity classes

(Connections between time, space and non-determinism)

- $TIME(f(n)) \subseteq NTIME(f(n))$, since each deterministic machine is also a non-deterministic one (by definition)
- $SPACE(f(n)) \subseteq NSPACE(f(n))$(as above)
- $TIME(f(n)) \subseteq SPACE(f(n))$

Basic relations among complexity classes

(Connections between time, space and non-determinism)

- $\mathrm{TIME}(f(n)) \subseteq \mathrm{NTIME}(f(n))$, since each deterministic machine is also a non-deterministic one (by definition)
- $\mathrm{SPACE}(f(n)) \subseteq \mathrm{NSPACE}(f(n))$(as above)
- $\mathrm{TIME}(f(n)) \subseteq \mathrm{SPACE}(f(n))$ (no machine can write more memory cells than its working time)
- $\mathrm{NTIME}(f(n)) \subseteq \mathrm{NSPACE}(f(n))$

Basic relations among complexity classes

(Connections between time, space and non-determinism)

- TIME($f(n)$) $\subseteq$ NTIME($f(n)$), since each deterministic machine is also a non-deterministic one (by definition)
- SPACE($f(n)$) $\subseteq$ NSPACE($f(n)$)(as above)
- TIME($f(n)$) $\subseteq$ SPACE($f(n)$) (no machine can write more memory cells than its working time)
- NTIME($f(n)$) $\subseteq$ NSPACE($f(n)$) (as above)
- NTIME($f(n)$) $\subseteq$ TIME($c^{f(n)}$) (by a theorem on simulating a non-deterministic machine by a deterministic one)

Gap Theorem and space-constructibility

Let assume the following constraint on f(n) (f(n) is space-constructible): $f(n) \geqslant \log n$ and there exists TM that, when receives n (in unary encoding) in input, uses exactly f(n) cells of space and stops

Example: $\lceil \log n \rceil, n^k, 2^n$ are space-constructible (and all "reasonable" functions are).

Gap Theorem and space-constructibility

Let assume the following constraint on f(n) (f(n) is space-constructible): $f(n) \geqslant \log n$ and there exists TM that, when receives n (in unary encoding) in input, uses exactly f(n) cells of space and stops

Example: $\lceil \log n \rceil, n^k, 2^n$ are space-constructible (and all "reasonable" functions are). (binary counter, multiplication/addition, n times doubling the input)

Theorem

(Gap theorem) There exists a recursive function f(n) so that TIME*(f(n))*=TIME*($2^{f(n)}$).*

Comment: constraints like space-constructibility are introduced to avoid situations like this.

Configurations of TM

The number of different configurations of a TM with space complexity f(n) (that is space-constructible) on a input word of length n can be **bounded by** $c^{f(n)}$, for some constant c that depends only on the machine and assuming that $f(n) \geqslant \log n$ (what is implied by space-constructibility)

$\mathrm{SPACE}(f(n)) \subseteq \mathrm{TIME}(c^{f(n)})$, due to the bound on the number of configurations ($c^{f(n)}$), since the machine that does not loop can be simulated on a machine that works that long (else it loops)

More relations between classes ...

- NTIME($f(n)$) $\subseteq$ SPACE($f(n)$), since deterministic machine can simulate non-deterministic one It suffices to generate each of f(n) sequences of non-deterministic choices (here we use the space-constructibility assumption) that are made during the computations. Next, we deterministically simulate the computations in f(n) steps. All these operations can be done in f(n) space since each sequence of non-deterministic choices can be simulated in the same space.
- NSPACE($f(n)$) $\subseteq$ TIME($c^{f(n)}$), again, due to simulation

  As before, the number of all configurations is $c^{f(n)}$, but now transitions between the configurations form a graph. It suffices to check whether there exists a path from the starting configuration to the terminating one, that can be computed in polynomial time (with regard to the graph size), that is in asymptotic time $c^{f(n)}$.

- L $\subseteq$ NL $\subseteq$ P $\subseteq$ NP $\subseteq$ PSPACE
- PSPACE $\subseteq$ NPSPACE $\subseteq$ EXP $\subseteq$ NEXP $\subseteq$ EXPSPACE $\subseteq$ NEXPSPACE

Explanations:

- NL $\subseteq$ P, due to NSPACE$(f(n)) \subseteq$ TIME$(c^{f(n)})$, because $c^{\log n} = n^k$
- NPSPACE $\subseteq$ EXP, also due to NSPACE$(f(n)) \subseteq$ TIME$(c^{f(n)})$, because $c^{n^k} = 2^{n^{k'}}$.

The space-hierarchy theorem (later) implies $L \subsetneq$ PSPACE so that the first and last elements above are different. Thus, at least one of the inclusions is sharp, however it is not known which one! (the most famous is the P vs NP case)

Theorem

(Savitch) If f(n) is space-constructible, then $\mathrm{NSPACE}(f(n)) \subseteq \mathrm{SPACE}(f^2(n))$.

Thus, we can infer that some classes are equal:

- PSPACE = NPSPACE,
- EXPSPACE = NEXPSPACE.

This means that non-determinism does not extend computational power for some high space complexity classes. Nothing comparable concerning the time complexity classes is known.

Space hierarchy theorem

Theorem

If f(n) is space-constructible and $g(n) \in o(f(n))$ (grows asymptotically slower) then $\mathrm{SPACE}(g(n)) \subsetneq \mathrm{SPACE}(f(n))$.

Time hierarchy theorem

f(n) is time-constructible iff $f(n) \geqslant n \log n$ and there exists a TM that having n (unary encoding) on input can work in exactly f(n) steps and halt. ("most" of known functions have this property)

Theorem

If f(n) is time-constructible and $g(n) \in o(f(n)/\log f(n))$ then $\mathrm{TIME}(g(n)) \subsetneq \mathrm{TIME}(f(n))$.

Conclusions

- $\mathrm{SPACE}(n^{\epsilon_1}) \subsetneq \mathrm{SPACE}(n^{\epsilon_2})$, for $0 \leqslant \epsilon_1 < \epsilon_2$, from the properties of polynomials
- $\mathrm{TIME}(n^{\epsilon_1}) \subsetneq \mathrm{TIME}(n^{\epsilon_2})$, for $1 \leqslant \epsilon_1 < \epsilon_2$, as above
- $\mathrm{L} \subsetneq \mathrm{PSPACE}$, since logarithm grows slower than polynomial
- $\mathrm{P} \subsetneq \mathrm{EXP}$, since each polynomial grows slower than sub-exponential function $n^{\log n}$ that grows slower than any exponential function
- $\mathrm{PSPACE} \subsetneq \mathrm{EXPSPACE}$, as above

Thus, the complexity class P that is usually regarded as the class of "efficiently solvable" problems has some inner hierarchy.

Literature

- Complexity Theory:
  - Papadimitriou "Computational Complexity", Addison Wesley Longman, 1994
  - Garey, Johnson "Computers and Intractability" (issued in 1979, difficult to get nowadays)
- Introductory Textbooks:
  - Cormen et al. "Introduction to Algorithms" 3rd edition, MIT Press
    chapters 34,35
  - Kleinberg, Tardos "Algorithm Design", Addison Wesley, 2006
    chapters 8,10,11

Thank you for attention