Reikniritt, rökfræði og reiknanleiki

Magni og Anna

skil 11

1 Exercise 8.1 bls 302

Show that for any function f: $N \to N$, where $f(n) \geq n$, the space complexity class SPACE($f(n)$) is the same whether you define the class by using the single-tape TM model or the two tape read-only input TM model.

The two-tape read-only TM model is used to allow sublinear space complexity classes to be defined. Since the input is read-only, there is no need to count the reading of the input as access to storage cells. However, when f(n) $\geq$ n, the amount of space used for the input is obviously linear in terms of the input size, and hence the number of storage cells that must be scanned in order to read the input is linear as well. So, when f(n) $\geq$ n, the cost of scanning across the input does not change the space complexity. And it is possible for the single-tape TM to operate in exactly the manner as the two-tape read-only input TM model by scanning across its input and using the cell in the n+1 position as the start of the storage tape, while using the first n cells as its read only input. Since this only adds a linear number of storage cells above what the 2-tape model would have used, and since f(n) $\geq$ n, the same space complexity class SPACE(f(n)) would be represented.

2 Exercise 8.4 bls 302

Show that PSPACE is closed under the operations union, complementation, and star.

A,B $\in$ PSPACE
L(M_1)=A, L(M_2)=B
M_1 og M_2 eru P-SPACE

Find M_3 L(M_3) = A $\cup$ B
ef við keyrum þær hvor eftir annari þá fáum við $f_1(n) + f_2(n)$ sem er P-SPACE
ef við keyrum þær samhliða þá fáum við 2 $* Max(f_1(n), f_2(n))$ sem er P-SPACE

star:
Finna M_3 L(M_3) = A*
Við skiptum orðinu í búta og stærsti búturinn verðu ekki stærri en upprunalega orðið. Þannig við þurfum ekki meira minni en lengdin á orðinu + smá constant sem er P-SPACE

complementation:
Tekur jafn mikið pláss
$\bar{f}(n) = f(n)$

3 Exercise 8.6 bls 302

Show that any PSPACE-hard language is also NP-hard

Given any PSPACE-hard language L we know by definition that every language L′ in PSPACE is polynonial time reducible to L. Since we know that NP $\subseteq$ PSACE, this means that every language L_{NP} in NP is in PSPACE and therefore polynomial time reducible to L. By definition of NP-hard, since every language L_{NP} in NP is plynomial time reducible to L, L is NP-hard.

3 Exercise 8.9 bls 303

Show that, if every NP-hard language is also PSPACE-hard, then PSPACE = NP

P $\leq$ NP $\leq$ P-SPACE
NP $\leq$ P-SPACE

Given every NP-Hard language is also PSPACE-Hard want to show that NP = PSPACE. From our assumption we know that if every NP-Hard is also PSPACE-Hard we know that then every NP-Complete language is also PSPACE-Hard since NP-Hard contains all of the NP-complete problems by definition. So we also know that SAT is PSPACE-Hard. And from the assumption that for any A in PSPACE, A reduces to SAT. Claim then we can solve A in NP. Create a TM, N as follows.

on input x, do
Compute f(x), the poly-time reduction between A and SAT.
Decide whether f(x) is satisfiable, if so, accept, otherwise reject.

Claim N decides A since x is in A iff f(x) is in SAT. Also notice that N is an NP machine since computing SAT is in NP.

4 Exercise 8.13 bls 303

Show that TQBF restricted to formulas where the part following the quantifiers is in conjunctive normal form is still PSPACE-complete.

TQBF $\in$ PSPACE sammkvæmt 8.8 bls 284
$\neg(\phi \vee \psi) = (\neg\phi) \vee (\neg\psi)$
$\neg(\neg p) = p$
$(\phi \wedge \psi) \vee x = (\phi \vee x) \wedge (\psi \vee x)$

5 Exercise 7.26 bls 274

We can place each card in the box in two possible orientations, the normal or the reverse orientation. We don't care about the relative positions of the cards in the box, that is we don't care if card A is on top of card B or the opposite, since in both cases the holes are covered in the same way.
First, we show that PUZZLE is in NP. We construct a nondeterministc polynomial time Turing machine M that decides PUZZLE as follows. Machine M on input $(c_1, ..., c_k)$ chooses nondeterministically an orientation for each of the cards c_i and places the cards in the box. Then it checks to see if all the holes are covered, and if they are covered it accepts, othewise it rejects. The running time of machine M is polynomial in the number of holes and the number of cards.
Second, we show that every language in NP is polynomial time reducible to PUZZLE. To do that, we show that 3SAT is polynomial time reducible to PUZZLE. That is, for each 3cnf-formula ϕ, we show how to construct, in polynomial time, specific cards for the PUZZLE problem, such that the PUZZLE has a solution iff ϕ is satisfiable.
We construct the cards from the 3cnf-formula ϕ in the following way.
For each variable x_i of ϕ we create a new card c_i.
In each card, we divide the two columns in row, such that each row corresponds to a clause of ϕ. (The number of rows is equal to the number of clauses of ϕ.)
Id card c_i, if literal x_i appears in clause j then we don't punch a hole in the left position of row j. Otherwise, if x_i doesn't appear in clause j then we punch a hole in the left position of j.
In card c_i, if literal $\neg x_i$ appears in clause j then we don't punch a hole in the right position of row j. Otherwise, if $\neg x_i$ doesn't appear in clause j then we punch a hole in the right position of row j.
We also create a special card that has holes in all its left column, and no holes in its right column.
To show that this construction works, we argue that if ϕ is satisfiable then the PUZZLE has a solution, and conversely, if PUZZLE has a solution then ϕ is satisfiable.

Suppose that ϕ is satisfiable. Then we can solve the PUZZLE as follows. Since ϕ is satisfiable, there exists an assignment to its variables that satisfies ϕ. For each variable x_i of ϕ, if x_i is 1, in the satisfyng assignment, we put the corresponding card c_i, in the box in its normal orientation, otherwise, if x_i is 0, we put the card in its reverse orientation. We also put the special card in its normal orientation. This placement of the cards covers completely the bottom of the box. Assume, for contadiction that this is not true, and therefore there exists a row j, that corresponds to clause j, through which you can view the left bottom of the box. For each card c_i, with normal orientation, we have that its corresponding variable x_i has a hole in the left position of row j, which means that the variable appears either as $\neg x_i$ or not at all in clause j. Since c_i has normal orientation, we have that $\neg x_i = 0$ and therefore variable x_i does not satisfy clause j. Thus the variables of the normal oriented cards do not satisfy clause j. Using a similar reasoning we have that the variables of the cards with the reverse orientation do not satisfy the clause j either. Thus the initial assignment of the variables does not satisfy the formula, which is a contradiction, since we have that the assignment is satisfying.

Suppose now that the PUZZLE has a solution. Then we show that ϕ is satisfiable, by constructing a satisfying assignment as follows. The special card, according to its orientation, does not cover one of the two columns. If it does not cover the left column, then for each row j there exists at least one card c_k that has its left position of row j with no hole. If the card c_k has normal orientation then

we assign the value 1 to its variable c_k, otherwise if the orientation is reverse we assign the value 0. We repeat the same procedure for all the rows. If by the above procedure some of the variables have not been assigned values, we assign to them an arbitrary value 0 or 1. The procedure is similar when the special card has reverse orientation. This assignment is a satisfying one, since for each clause j (that corresponds to row j) there exists at least one variable whose literal x_i or $\neg x_i$ in the clause j has value 1. If the literal is x_i then its card c_i has normal orientation, otherwise if the literal is $\neg x_i$ then the its card c_i has reverse orientation, when the special card has normal orientation.
The above reduction takes polynomial time in the number of clauses and variables of a 3cnf- formula ϕ. This completes the proof.

6 Exercise 8.14 bls 303