

Xiaoyun Ren

CMPS-C464

Collaborator: None

1. In the proof of the Cook-Levin theorem, show why fail if use 2×2 windows instead of 2×3

Because during the proof, we need to check the validity of the window

if use 2×2 window instead

$$\begin{array}{|c|c|} \hline t_1 & \\ \hline t_0 & \\ \hline \end{array} \text{ or } \begin{array}{|c|c|} \hline t_0 & t_1 \\ \hline & t_1 \\ \hline \end{array} \text{ instead of } \begin{array}{|c|c|c|} \hline t_1 & t_2 & t_3 \\ \hline t_0 & t_1 & t_2 \\ \hline \end{array} \text{ or } \begin{array}{|c|c|c|} \hline t_2 & t_3 & t_1 \\ \hline t_1 & t_2 & t_3 \\ \hline \end{array}$$

2×2 windows are always valid while 2×3 windows are not

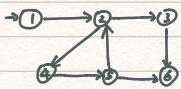
therefore, using 2×2 windows in proof of Cook-Levin Theorem will fail the proof.

reference: course.cs.washington.edu/courses/cse431/14sp/lectures/lec12.pdf

isical.ac.in/~marijith/Courses/spring2010/slides/complexity/lec6.pdf

carlisle.andrew.cmu.edu/~mofekhaf/15-815/lectures/lecture7.pdf

2.



Both player I and II have a winning strategy.

Win strategy for player I:

player I: Start at Node 1, choose Node 2

player II: Pick from Node 4 or 5, choose Node 3

player I: Only 1 choice: Node 6, choose Node 6

player II: Got stuck in Node 6

for player II:

player I: Start at Node 1, choose Node 2

player II: Pick from Node 4 or 5, choose Node 4

player I: Only 1 choice: Node 5, choose Node 5

player II: Only 1 choice: Node 6, choose Node 6

player I: Got stuck in Node 6

3. To show A_{LBA} is PSPACE-Complete

① A_{LBA} is in PSPACE

thus, construct a deterministic TM M_1 to decide A_{LBA} in poly-space

say M is an LBA with q states and g symbol, $n=|w|$

M_1 = an input $\langle M, w \rangle$ where M is an LBA

i) Simulate M on w for g^{n^2} steps or until it halts

ii) if M halt, if M accept, accept, if reject, reject

iii) if M doesn't halt, reject

if M doesn't halt in g^{n^2} steps $\Rightarrow M$ runs out of n -bounded tape

with M_1 , running in poly-space, since g^{n^2} assumes the value is at most exponential of length $|w|$

we can say M_1 decides A_{LBA} in poly-space, A_{LBA} in PSPACE

② every A in PSPACE is poly-time reducible to B (A_{LBA} is PSPACE-hard)

assume language L in PSPACE, there's a TM M_2 decides L

$L \leq_p A_{\text{LBA}}$ by give a mapping function f maps w onto $\langle M, w \rangle_{L_2}$

f is polytime reduction from L into A_{LBA}

Therefore, with ① and ②, A_{LBA} is PSPACE-Complete

reference: people.csail.mit.edu/rivluo/solving/soln.ps
people.cs.ann Arbor/courses/slides-CC-10.pdf

4.

a) Show that MIN-FORMULA is PSPACE

MIN-FORMULA is a collection of boolean formulas where is minimal

$$\Rightarrow \text{MIN-FORMULA} = \{\phi \mid \phi \text{ is minimal}\}$$

design an algorithm f for MIN-FORMULA in poly-space

f is on input ϕ , ϕ is a boolean formula

i) Simulate boolean formulas b where b is smaller ϕ

ii) if b and ϕ not equivalent, accept, else reject

Since, string b takes at most $|\phi|$ space, storing all variable assignments takes at most $|\phi|$ space

and evaluating b or ϕ on an assignment takes at most $|\phi|$ space

we say this algorithm of is poly-time, therefore MIN-FORMULA in PSPACE

reference: people.csail.mit.edu/rw/6.045-2020/ec17-color.pdf

b)

ADD: $x, y, z \geq 0$ are binary integer, $x+y=z$?

to show ADD $\in L = \text{PSPACE}(\log n)$ n is bits of binary x, y, z

design an algorithm f for ADD in log space

assume $n=2k$, k is an integer, if n is odd, add 0 to the front, ex: 111 \rightarrow 0111

i) encode x, y into $x_0x_1\ldots x_{k-1}, y_0y_1\ldots y_{k-1}$

ii) divide and conquer: partition into 2 parts with length $\frac{n}{2}$
:

iii) addition $\rightarrow x_0y_0, \dots, x_{k-1}y_{k-1}$ where $z_i = x_i + y_i$

iv) output z_i

looks like it takes up to 2^n space, but no, with divide and conquer, store one of partitions at a time

instead we use space at most $\log n$

therefore, ADD $\in L = \text{PSPACE}(\log n)$

reference: stackoverflow.com/questions/32926732/how-does-binary-sum-use-o(logn)-space

math.stackexchange.com/questions/1341/lecture/6.htm