Xiaoyu Pen CMPSC464 Collaborator: None

1. In the proof of the Cook-Levin theorem, show why fail if use 2x2 windows instead of 2x3

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

if use 2x2 window instead

2x2 windows are always valid while 2x3 windows are not

therefore, using 2x2 windows in proof of Cook-Levin Theorem will fail the proof.

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

isical.ac.in/~arijit/courses/spring2010/slides/complexity/lec6.pdf

contrib.andrew.cmu.edu/~ofakaha/ffiac-s15/lectures/Lecture17.pdf

2.

Both player I and II have a wining strategy

Win strategy for player I:

player I: Start at Node 1, choose Node 2

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

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

player II: Get stuck in Node 6

for player II:

player I: Start at Node 1, choose Node 2

player II: Pick from Node 4 or 3, 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: Get 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 = on input $\langle M, w \rangle$ where M is an LBA

1) Simulate M on w for qng^n steps or until it halts

2) if M halt, if M accept, accept, if reject, reject

3) if M doesn't halt, reject

if M doesn't halt on in qng^n steps $\Rightarrow$ M runs out of n-bounded tape

with M_1 running in poly-space, since qng^n 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, then's a TM M_2 decides L

$L \leq_p A_{LBA}$ by give a mapping function f maps w onto $\langle M, w \rangle L_c^{|w|}$

f is poly-time reduction from L into A_{LBA}

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

reference: people.eecs.berkley.edu/~luca/cs172/sol11.ps

people.cs.aau.dk/~srba/courses/slides-CC-10.pdf

4.

a) Show that MIN-FORMULA in PSPACE

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

=> MIN-FORMULA = {ϕ | ϕ is minimal}

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

f= on input ϕ, ϕ is a boolean formula

1) Simulate boolean formulas b where b is smaller ϕ

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

Since, storing b takes at most $|\phi|$ space, storing an variable assignment takes at most $|\phi|$ spac
and evaluating b or ϕ on assignment takes at most $|\phi|$ space

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

reference: people.csail.mit.edu/rrw/6.045-2020/lec19-color.pdf

b)

ADD= {<x,y,z> | x,y,z >= are binary integer, x+y=z}

to show ADD $\in$ L, L=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 -> 0111

1) encode x,y into $x_n x_{n-1} \dots x_1, y_n \dots y_1$

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

⋮

3) addition -> $x_1+y_1, \dots, x_n+y_n$ where $z_i = x_i + y_i$

4) output z_i

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

instead we use space at most log n

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

reference: stackoverflow.com/questions/32926732/how-does-binary-sum-use-olgn-space

mathcounts.com/courses/banning/341/lecture/6.htm