1a. These are definitions

1bi. If L’ is in NP, then there exists an NDTM M’ deciding L’ in polynomial time. Since L ≤ L’, there exists a reduction function f which is computable by a DTM in polynomial time. Construct an NDTM M which, given input x, first computes f(x) deterministically and then runs M’ with input f(x). If f is computable is O(n^a) and M’ is O(n^b), then |f(x)| is bounded above by O(n^a), so M’ runs in O(n^ab) time. This is still polynomial, so M runs in p-time and L is also in NP.

1bii. Denote by –L the complement of a language L (word online is bad). If L ≤ L’, then –L ≤ -L’. It follows that if a class is closed downwards under reduction, then so is its complementary class.

1c. Suppose L’ is a language in NP. Then L’ ≤ L since L is NP-hard. So then, -L’ ≤ -L. Since L is in co-NP, it follows that –L is in NP. So –L' is in NP since NP is downwards closed. Hence L’ = -(-L’) is in co-NP. Conversely, suppose L’ is a language in co-NP. Then -L’ ≤ L since –L' is in NP. So then L’ ≤ -L. But then L is in co-NP so –L is in NP. Hence L’ is in NP. Therefore co-NP = NP.

1d. HPNE is NP: given G and e, guess a node x of G and a path of length n beginning at x. Then verify by following the path that it does not use e and visits every node. These are both polynomial time operations.

HPNE is NP-Hard: we show HP ≤ HPNE. Let G be an input to HP. Choose any edge (x, y) in G and duplicate the edge. The duplicated edge is e and the new graph is G’. This is clearly p-time. If G has a Hamiltonian path, then this path is also a Hamiltonian path in G’ which doesn’t use e, since e is not in G. Conversely, if G’ has a Hamiltonian path not using e, then this is a Hamiltonian path in G since G is G’ \ {e}.

2a. Definitions

2b. BDDRCH is NL. Use the same NDTM as for RCH: follow a path from x, keeping a counter of the length. Stop if we reach y, or if the counter hits k. If we reach y, then return ‘yes’, otherwise ‘no’. This uses log(n) space for the counter, where n is the number of nodes.

BDDRCH is NL-complete since RCH logspace reduces to BDDRCH: given an input (G, x, y) to RCH, it is a yes instance if and only if (G, x, y, n) is a yes instance of BDDRCH.

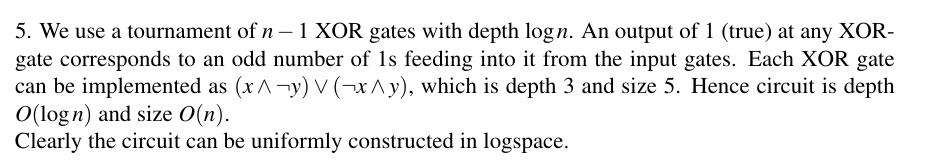
2c. A cycle of odd length in G which repeats a node y can be decomposed into two cycles starting (and ending) at y. One of these two cycles must have odd length. So, to find cycles of odd length, it suffices to consider cycles which don’t repeat nodes, and hence have length at most n (n = # of nodes in G).

Build an NDTM as follows: choose a node x and follow a path starting at x, keeping a counter of the length. Stop if the counter goes above n. If we reach x again, then check whether the cycle was odd length. If so, return ‘yes'. Return ‘no’ in all other cases. This uses log(n) space to store x and log(n) space for the counter. This machine decides –2COL, so –2COL is in NL, and 2COL is in co-NL. But NL = co-NL, so 2COL is in NL.

2d. Consider the following NDTM. Choose an edge e = (x, y) in G. If there are multiple edges from x to y, then return ‘yes’. Otherwise, start at x and follow a path from x keeping a counter of the length and making sure to not use e at any point. If the counter goes above n, then stop. If we reach y, stop and return ‘yes’. In all other cases return ‘no’. This is logspace (uses O(log n) to store the edge e = (x, y) and log(n) for the counter). It decides –BRIDGE. Similarly to above, it follows that BRIDGE is in NL.

3a. Definitions

b. On problem sheets.



ci. ALLONES is in AC0. For any n, we have a uniform circuit consisting of n inputs feeding into a single and-gate. This is constant depth and linear in size, so it is in AC0.

ALLONES is *not* in NC0. Suppose there was a uniform family of circuits of depth bounded above by a constant c. To decide whether a word w is in ALLONES, clearly every bit in the input must be looked at by the circuit, I.e. every bit is the input to some gate. A boolean circuit of depth c can have at most 2^c inputs. So then the circuit for n = 2^{c + 1} can consider at most 2^c inputs, and hence cannot decide whether a word w with |w| = n is in ALLONES.

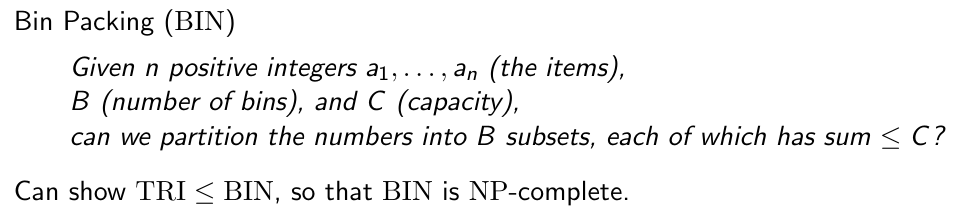
cii. If A = (a\_ij), B = (b\_ij) are two n by n boolean valued matrices, and C = AB, then the entries of C are given by:

c\_ij = (a\_i1 ∧ b\_1j) ∨ … ∨ (a\_in ∧ b\_nj)

To compute this, we require n and-gates and a single or-gate with n inputs. Hence the circuit to compute matrix multiplication has constant depth 3 (input layer, layer of and-gates and a layer of or-gates) and is polynomial in size. So boolean-valued matrix multiplication is in AC0. So to compute RCH, we can square the adjacency matrix O(log n) times to obtain the transitive closure, and then read off the answer. This amounts to sequencing the circuit for matrix multiplication with itself log(n) times, so RCH is AC1.

4a. Moar definitions

b.



bii. Given an oracle for BIN, we can solve MINBIN. First iterate through the a\_j to check that they are all ≤C. If not, return ‘no’. Otherwise, we have 1 ≤ B ≤ n. invoke BIN with B = 1, 2, …, n. This is linearly many calls to the oracle. Stop and return yes the first time the oracle returns ‘yes’.

Biii. No. If MINBIN is in FNP, then the relation “R({a\_1, …, a\_n, C}, B) if and only if B is the smallest number such that we can partition {a\_1, …, a\_n} into B subsets each with sum ≤ C” is poly decidable. But then we have an algorithm for BIN, by letting B = n. This implies that P = NP.

4c. In problem sheets.

