2020 cmput 304 final

1. Two sets *interfere* if they have 4 or more common elements. This is problem NICS: given integers n and k and a collection C = {S0, S1, ...} of subsets of integers {0, 1, ... n}, is there a collection T of exactly k subsets from C, such that each pair of subsets Sx, Sy in T is non-interfering?

E.g. for the problem at **right**, {S1, S2, S4} is a NICS with $k = 3$: S1 and S2 do not interfere (3 elements in common), S1 and S4 do not interfere (0 elements in common) S2 and S4 do not interfere (2 elements in common).

```
    0 1 2 3 4 5 6 7 8 9
S0  * * - * - * * - *
S1  * * * - * - - * *
S2  * - * - * * * - *
S3  - * - - * - - * *
S4  - * * * - * - * *
```

```
    0 1 2 3 4 5 6 7 8
S0  - * * * - * - * *
S1  * * - * - * * - *
S2  * * * - * - - * *
S3  * - * - * * * - *
S4  - - * - - - - * -
```

A) **(3 marks)** For the collection above **left**, give a NICS with $k = 3$ or explain briefly why there is none.

Answer only one of B) and C). (C is hard: for 5 marks instead of 12, answer B instead of C.) (If you answer both B and C, we will ignore your answer to C and mark only B.)

B) **(5 marks)** Prove that problem NICS is in NP.

C) **(12 marks)** Assume that problem NICS is in NP.

Prove or disprove: problem NICS is NP-complete.

2. Below is code (with some print statements removed) from class github repo iterative backtracking sat-solver `backsat-v2.py`. **Answer only one of A) and B).** (B is hard: for 10 marks instead of 15, answer A instead of B.) (If you answer both A and B, we will ignore your answer to B and mark only A.)

A) **(10 marks)** Give empirical evidence (e.g. generate a sequence of inputs, and track the number of iterations)

that worst-case runtime for `backsat(f,a,v)`

when input is a 2-sat formula is polynomial . . . or . . .

that worst-case runtime for `backsat(f,a,v)`

when input is a 2-sat formula is super-polynomial.

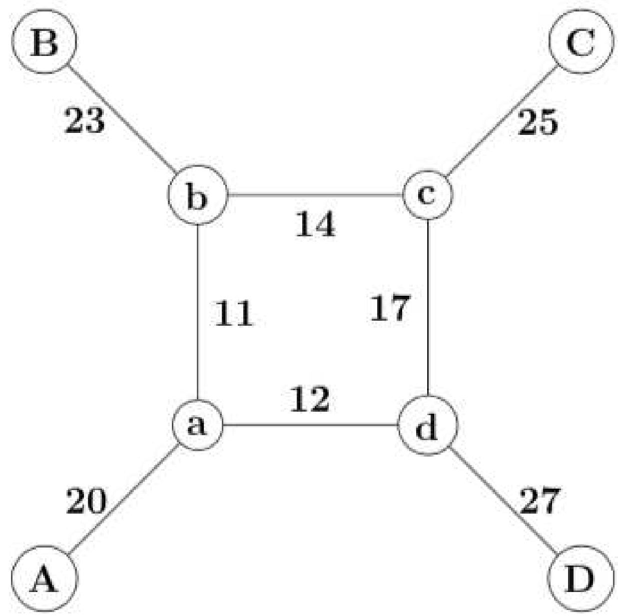
B) **(15 marks)** Prove or disprove:

worst-case runtime for `backsat(f,a,v)`

when input is a 2-sat formula is polynomial.

```
def backsat(f, a, v): # formula, assignment, verbose
  itns, candidates = 0, []  # list of partial solutions
  while True:
    itns += 1
    if sat(f):
      return f, a
    if unsat(a):
      if len(candidates) > 0:
        f, a = candidates.pop()
      else:
        return f, a, itns #
    else: # f != empty list, a != empty string
      ndx = ind_short(f)
      lenj = len(f[ndx])
      if lenj == 1: # fix literal
        f, a = fix_literal(f[ndx][0], f, a, v)
      elif lenj >= 2: #try both possible values
        fcopy, acopy = deepcopy(f), a
        f, a       = fix_literal(f[ndx][0], f, a, v)
        newf, newa = fix_literal(-fcopy[ndx][0],fcopy,acopy,v)
        candidates.append((newf, newa))
```

3. You manage a communications network with users A,B,C,D and bandwidths shown in the figure below. You need to establish a connection between each pair of users except for B and D (they never communicate). Connections A-B, A-C, A-D, B-C, C-D, pay 5, 2, 3, 1, 4 dollars respectively per unit bandwidth. Between each pair of users, at least 3 units must be routed.



There are two possible routes for every connection. For connection A-B, let xAB be the traffic volume routed A-a-b-B (the short way) and yAB the volume routed A-a-d-c-b-B (the long way). Define xBC, yBC, xCD, yCD, xAD, yAD similarly. Let xAC be the volume routed A-a-b-c-C and yAC the volume routed A-a-d-c-C.

You want to maximize this network's revenue. Using the variables defined above, formulate this problem as a linear program. **You do not need to find an optimal solution.**

A) Give the objective function.

B) Give the system of inequalities.

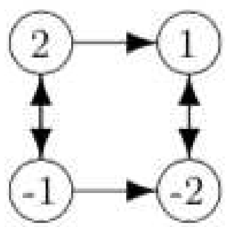
C) Give a feasible solution.

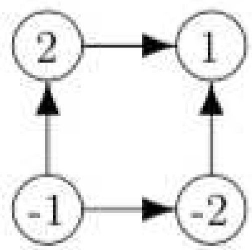
4. A. Give the dual of this linear program (LP):

$$\begin{array}{rrcrcrcr}
\max & 4x_1 & + & 3x_2 & + & x_3 & & \\
\text{s.t.} & & & x_2 & + & 3x_3 & \leq & 5 \\
 & 2x_1 & & & - & 4x_3 & \leq & -1 \\
 & x_1 & + & x_2 & + & x_3 & \leq & 6
\end{array}$$

B. Give an easily verified proof that the optimal value of the LP is at most 16.

5. In short form, we write 2-sat formula f $(x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2)$ as `[1 2][1 -2][-1 -2]`. The implication digraph D(f) (below **left**) has two strongly connected components (SCCs): `X={-2,1}` and `Y={-1,2}`. An ordering $(C_0, C_1, \ldots)$ of the SCCs of a digraph is *good* if, for each pair of indices $j < k$, there are no implications from any literal in C_j to any literal in C_k. E.g. for D(f), the ordering (X,Y) is good — there are no implications from X to Y — but the ordering (Y,X) is not good — there is an implication from Y to X. (X,Y) is the only good ordering for D(f). The implication digraph D(g) (below **right**) has two good orderings: `({1},{-2},{2},{-1})` and `({1},{2},{-2},{-1})`.





A) For formula h below, give a good ordering of D(h). (Within each SCC, write elements in sorted order, from smallest integer to largest integer.) **Explain briefly.**

```
h = [-2 -4][-2 -1] [-2 1][-6 -8] [-8 5][-3 1]

    [-7 5][7 3] [7 6][8 2][6 4]
```

B) Give the number of good orderings of D(h). **Explain briefly.**

C) Give

- either a satisfying assignment of values T/F to variables $x_1 \ldots x_8$ of h, written like this: T F F T F F T F. **Explain briefly.**
- or a variable t — so, from 1 to 8 — and an SCC of D(h) that includes both `t` and `-t`. **Explain briefly.**