Zadanie 5 - PIZZO

PIOTR SALAMON

Link do rozwiązań zadań: https://colab.research.google.com/drive/1ysNqU5illH-hxSdqjNC_DEJ81c20Jz5I?usp=sharing

Przygotuj pojedynczy plik PDF z raportem z wykonania tego zadania. W nim opisz szczegółowo swoje rozwiązanie części A, F, G, H i I. Oprócz tabel i wykresów, należy wyciągnąć wnioski z eksperymentów.

Stwórz program, który dla zadanej liczby n tworzy losowy graf o n wierzchołkach. Zadbaj o to, żeby generowane grafy były zróżnicowane.

```
def G(n, p):
    matrix = [[1 if (random.uniform(0, 1) <= p) else 0 for j in range(n)] for i in range(n)]
    e = sum([sum(v) for v in matrix])

    for i in range(n):
        for j in range(i + 1, n):
            matrix[j][i] = matrix[i][j]
    return matrix, e
```

Graf losowy generowany jest za pomocą modelu Erdős–Rényi (link).

Metodę generowania grafu można opisać następująco:

> Dla danych parametrów n (ilość wierzchołków) oraz p (prawdopodobieństwo wybrania krawędzi) stwórz macierz sąsiedztwa wielkości n x n. Każdą kolejną komórkę niezależnie uzupełnij wartością 1 z prawdopodobieństwem p.
>
> Generuję grafy nieskierowane więc następnie wymuszam symetryczność macierzy.

Pozwalam na pętle w grafie, nie wymagam spójności.

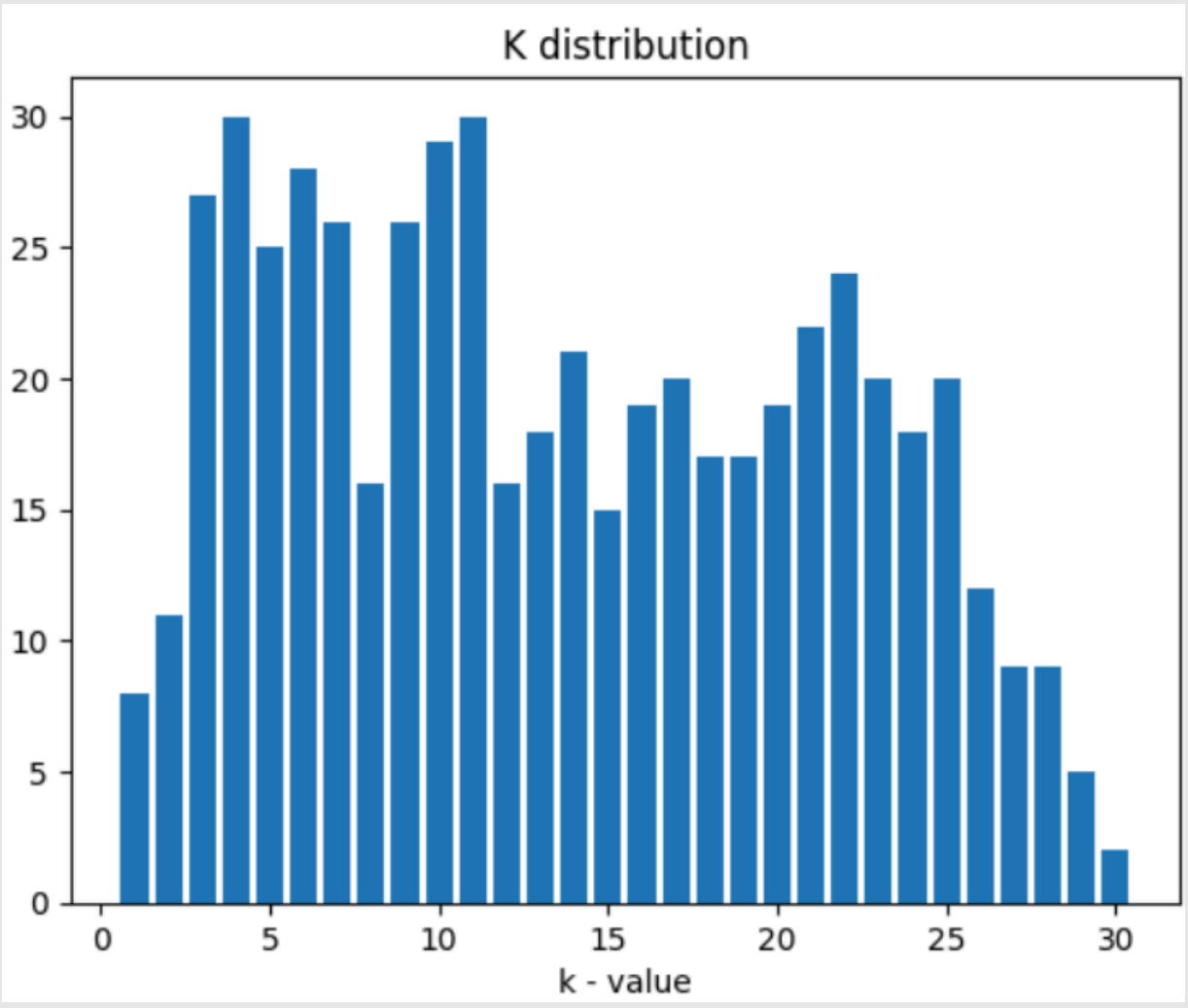
Stwórz program, który rozwiązuje problem vertex-cover z wykorzystaniem smt-solvera

```
def smt_solver(matrix, k):
    n = len(matrix)
    solver = z3.Solver()

    vertexes = [z3.Bool(v) for v in range(n)]

    for i in range(n):
        for j in range(i, n):
            if matrix[i][j]:
                solver.add(z3.Or(vertexes[i], vertexes[j]))

    solver.add(Sum([If(v, 1, 0) for v in vertexes]) <= k)
    return solver.check() == z3.sat
```

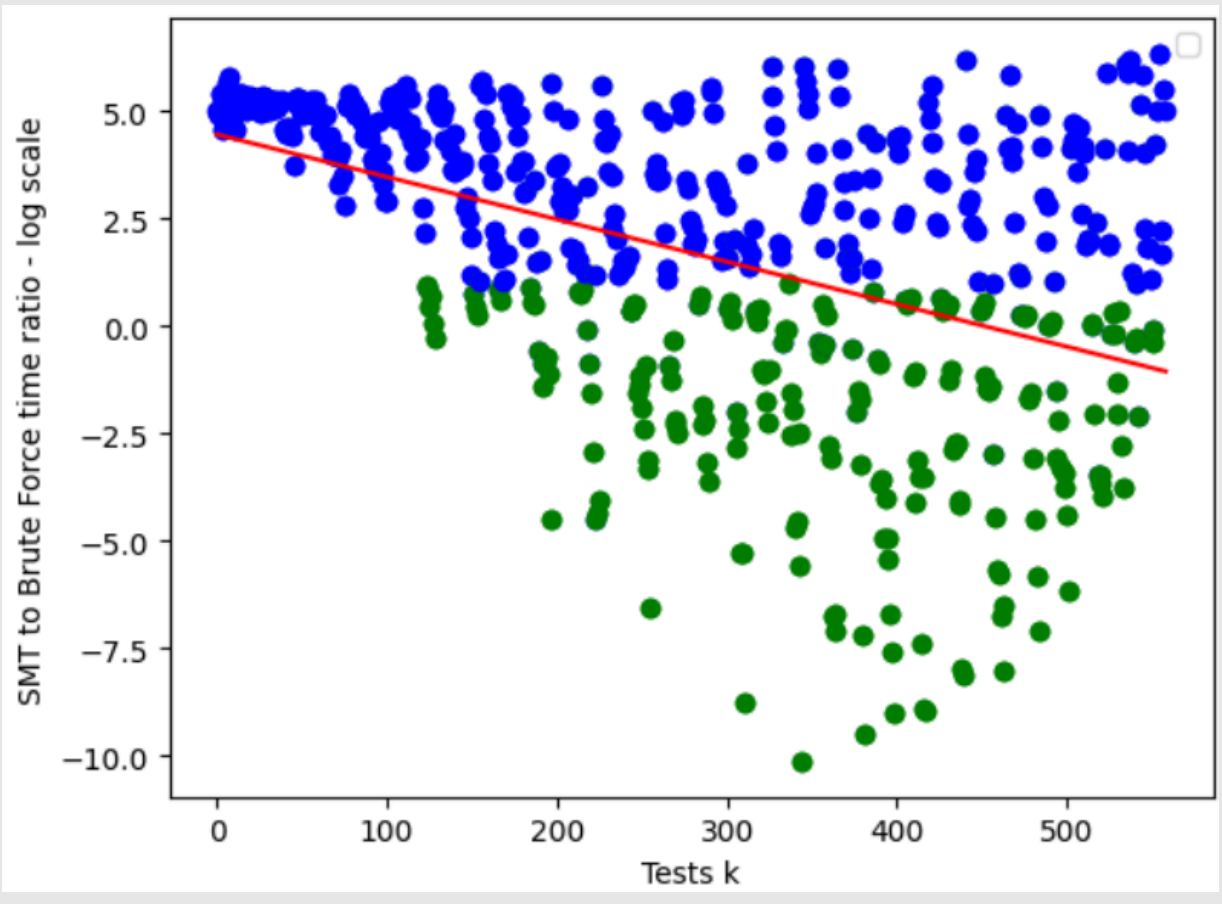
Dla każdego wierzchołka tworzę zmienną boolowską. Dla każdej krawędzi w grafie tworzę alternatywę zmiennych odpowiadającym obu jej końcom. W ten sposób wymuszam, że przynajmniej jeden koniec każdej krawędzi musi zostać wybrany. Następnie wymuszam na modelu, aby akceptował tylko takie wartościowania, które posiadają maksymalnie k zmiennych prawdziwych (jeżeli zmiennych będzie mniej niż k to można dopełnić pokrycie dowolnymi wierzchołkami do wielkości k)

Przeprowadź miarodajne eksperymenty porównujące czasy działania wszystkich stworzonych programów na grafach wygenerowanych w części C. Jak się zmieniają te czasy w zależności od rozmiaru grafu, a jak w zależności od liczby k?

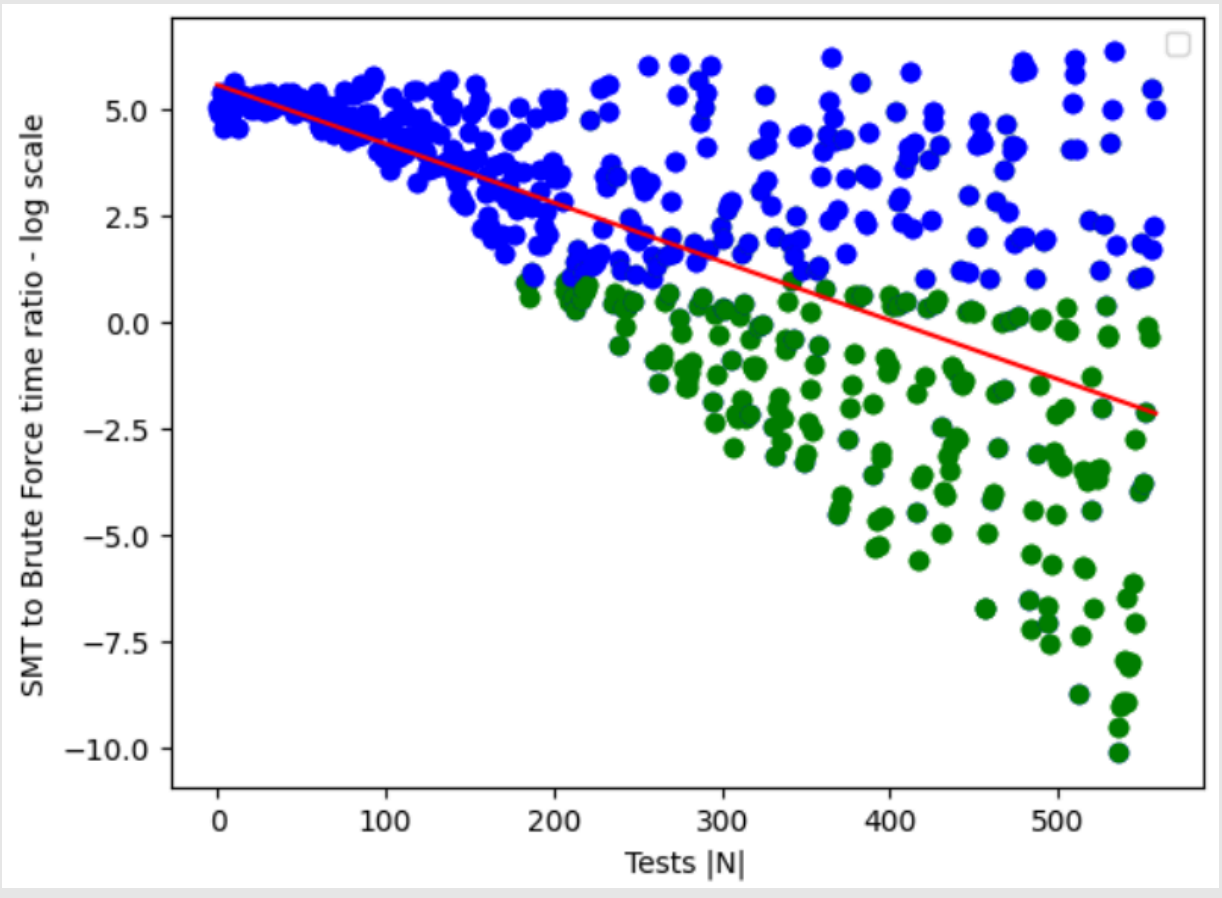
Do porównania algorytmów wykorzystałem stosunek czasów ich działania. Dla danego wejścia (G - graf, k - badany vertex cover) wyliczam t1 - czas działania rozwiązania korzystającego z SMT-solvera, t2 - czas działania brute-force. Wszystkie testy porównuję na podstawie wartości t1/t2.



Algorytmy były testowane na grafach losowych o takim rozkładzie k:



Dla rosnącego k średni stosunek zmniejsza się, a więc dla im większe k tym średnio szybciej wykona się program bazujący na SMT-solverze.



Dla rosnącego |N| średni stosunek zmniejsza się, a więc dla im większe |N| tym średnio szybciej wykona się program bazujący na SMT-solverze.

Wnioski:

Dla grafów małych i o małym k średnio lepiej sprawdzi się metoda brute-force. Małe k powoduje względnie małą ilość podzbiorów do rozważenia, a dla grafów małych czas inicjacji smt-solvera trwa na tyle długo, że niweluje to jego szybkość działania.

Dla grafów dużych i o większym k średnio lepiej sprawdzi się smt-solver. Przy większych czasach działania inicjacja solvera nie ma tak istotnego wpływu na całościowy czas działania.

Dla każdego programu eksperymentalnie określ, jak duże grafy i liczby k jest w stanie przetworzyć w dwie minuty

Ze względu na olbrzymią rozbieżność wyników w działaniu algorytmów (nawet dla bardzo bliskich sobie n i k) jako warunek konieczny uznania, że algorytm jest w stanie przeliczyć graf wielkości n w 120 sekund jest średni czas działania dla 5 próbek o stałej wartości n oraz p kolejno [10, 30, 50, 70, 90] mniejszy niż 120 sekund.

Dla algorytmu brute-force największe zaakceptowane n to 33 z czasami działania dla próbek: [366.s, 9.6s, 0.7s, 0.02s , 0.007s]

Dla algorytmu korzystającego z SMT-solvera największe zaakceptowane n to 112 z czasami działania dla próbek: [0.1s, 0.1s, 0.2s, 0.3s, 0.4s]

Ciekawy wykres nie będący wymagany w zadaniu: