Projekt "Rekonstrukcja" - dokumentacja końcowa

Filip Mieszkowski

Kurzątkowski Stanisław

Maj 2025

1 Wstęp

Niniejsza dokumentacja stanowi uzupełnienie głównej dokumentacji projektu "Rekonstrukcja". Zawiera ona listę zmian jakie zaszły w projekcie, które nie zostały opisane w dokumentacji głównej, a także opis działania programu dla użytkownika. Dodatkowo zapisane zostały wnioski, a także podział pracy.

2 Opis działania dla użytkownika

Cały projekt składa się z dwóch głównych plików: ***solve.py*** oraz ***create_instance.py***, dwóch dodatkowych plików: ***test_complexity.py*** oraz ***valid_check.py***, a także jednego pliku pomocniczego ***utils.py***. Plik ***create_instance.py*** służy do generowania przykładów, zaś ***solve.py***, stanowiący najważniejszy plik projektu, dokonuje rekonstrukcji drzewa.

2.1 Tworzenie instancji problemu

Program ***create_instance.py*** uruchamiany jest z trzema dodatkowymi parametrami:

- liczba wierzchołków w generowanym grafie
- ścieżka do wygenerowanego grafu (rozwiązania)
- ścieżka do wygenerowanej instancji problemu

Ponieważ drzewo nie może posiadać węzłów stopnia 2 (wyjaśnienie w głównej dokumentacji), to nie można jako liczby wierzchołków podać liczby 3. Użytkownik zostanie dodatkowo poinformowany o czasie, jak algorytm potrzebował na wygenerowanie tej instancji problemu.

2.2 Działanie głównego programu

Program ***solve.py*** uruchamiany jest z dwoma dodatkowymi parametrami:

- ścieżka do instancji problemu
- ścieżka do wygenerowanego rozwiązania

Dodatkowo program poinformuje o czasie działania algorytmu.

2.3 Walidacja

Program ***valid_ check.py*** uruchamiany jest z dwoma dodatkowymi parametrami:

- ścieżka do pierwszego grafu (np. wygenerowanego przy tworzeniu instancji)
- ścieżka do drugiego grafu (np. wygenerowanego poprzez rekonstrukcję drzewa)

Program ten sprawdza czy oba pliki reprezentują ten sam graf. Programy ***create_ instance.py***, ***solve.py*** oraz ***valid_ check.py*** można uruchamiać w dowolnej kolejności, jednakże poniższe rozważania na temat złożoności zakładają, że uruchamiamy je w kolejności podanej powyżej.

3 Zależność czasowa algorytmu — eksperymenty

W poprzedniej dokumentacji algorytmu zawarta jest obszerna sekcja, w której autorzy podjęli się dyskusji dotyczącej złożoności czasowej poszczególnych etapów algorytmu. Rezultatem była zamieszczona poniżej tabela:

Etap	**Złożoność czasowa**
Generowanie instancji	$\mathcal{O}(n^2)$
Rozwiązywanie	$\mathcal{O}(n^2)$
Walidacja	$\mathcal{O}(n^2)$
Całkowita złożoność	$\mathcal{O}(n^2)$

Table 1: Złożoność czasowa poszczególnych etapów algorytmu

Najciekawszym etapem algorytmu, jakim według autorów jest faza rozwiązywania instancji, towarzyszyły trudności w określeniu dokładnej złożoności. Końcowo przyjęto, że pesymistyczna złożoność tej części to $\mathcal{O}(n^2)$, aczkolwiek istotna część tego etapu ma złożoność $\mathcal{O}(n \log(n))$. W celu poszerzenia wątku dotyczącego złożoności czasowej, w tym raporcie autorzy zamieszczają analizę danych uzyskanych na drodze badania czasowej złożoności algorytmu. Poniższe wykresy powstały poprzez generowanie grafów o zwiększającej się o 100 liczbie wierzchołków. Dla każdego rozmiaru generowanych było 5 grafów, w celu wyznaczenia średniej i możliwości oszacowania niepewności punktu pomiarowego.

Dla każdego grafu rejestrowany był czas egzekucji poszczególnych etapów $(1-3)$ całej procedury.

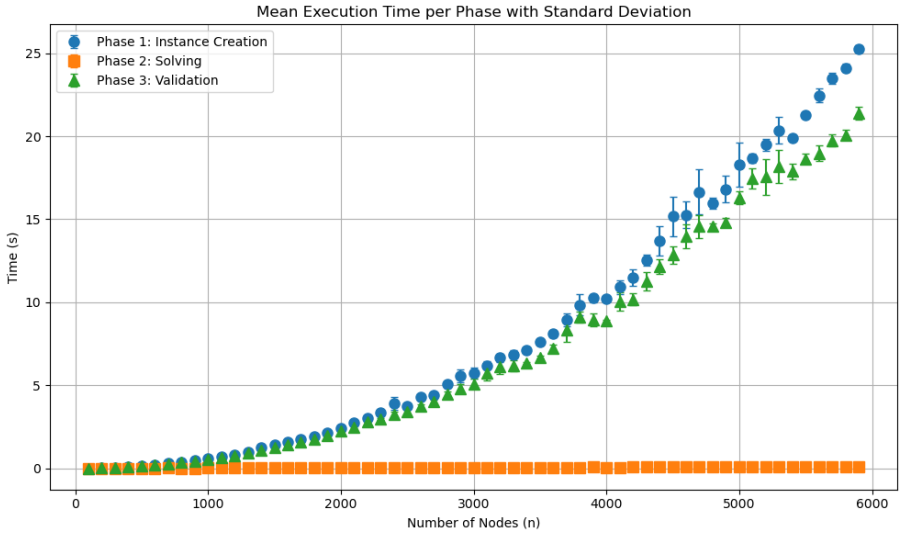

Figure 1: Udział każdego etapu algorytmu w złożoności czasowej

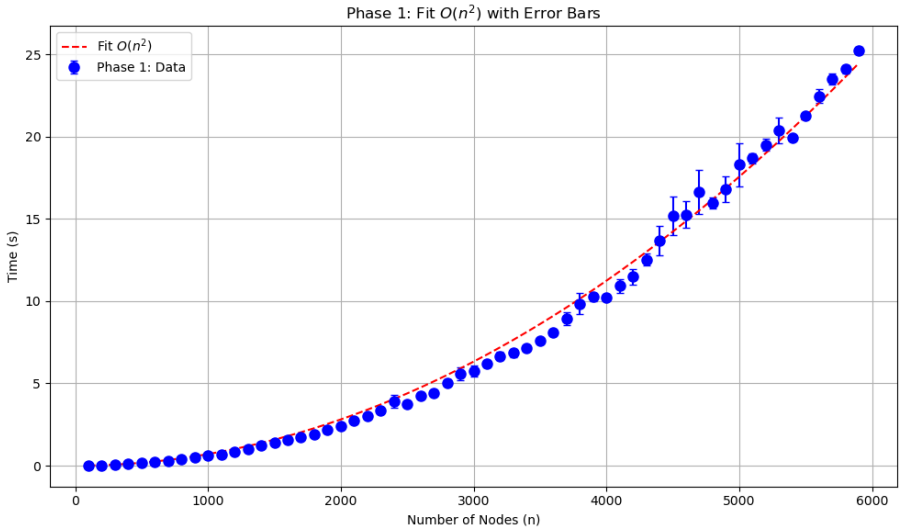

Figure 2: Dopasowanie wyrażenia postaci $a \cdot n^2$ do fazy generowania instancji

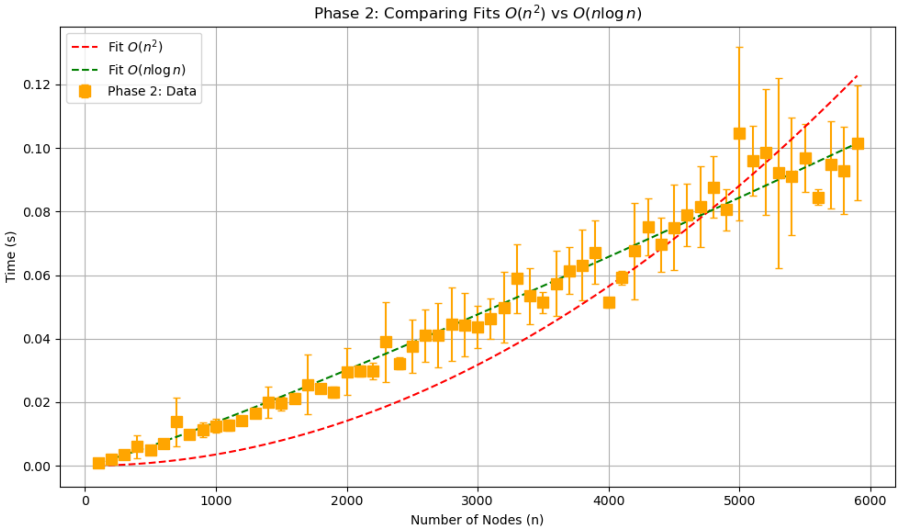

Figure 3: Dopasowanie wyrażeń postaci $a \cdot n^2$ oraz $a \cdot n \log(n)$ do fazy rozwiązywania

Figure 4: Dopasowanie wyrażenia postaci $a \cdot n^2$ do fazy walidacji

4 Wnioski

Wykres przedstawiający względny udział wszystkich 3 etapów algorytmu wyraźnie wskazuje na to, że etapy 1 i 3 faktycznie, zgodnie z teoretycznymi przewidywaniami, posiadają złożoność $\mathcal{O}(n^2)$. Odstaje od nich jednak etap 2 - rozwiązywanie, który cechuje się znacznie niższymi czasami. Wykresy o numerach $2-4$ prezentują dopasowania krzywych danego rzędu do punktów pomiarowych. Dla wykresów 2 oraz 4 dopasowane wyrażenia kwadratowe nie budzą zastrzeżeń, dlatego przyjęte w Tabeli 1 złożoności dla etapów ***generowania*** oraz ***walidacji*** można uznać za prawdziwą. Dla wykresu numer 3 odpowiadającego etapowi ***rozwiązywania*** dopasowane zostały dwie krzywe - $a \cdot n^2$ oraz $a \cdot n \log(n)$. Punkty pomiarowe, które przedstawiają średnie czasy potrzebne na zrealizowanie etapu algorytmu, odpowiadają raczej "oczekiwanej" złożoności ze względu na wyliczanie średniej. Z tego powodu wykres ten nie może stanowić argumentu za odrzuceniem złożoności $\mathcal{O}(n^2)$, jako złożoności pesymistycznej tego etapu. Może on natomiast stanowić przesłankę na temat tego, że złożoność oczekiwana tego etapu jest zbliżona do $\mathcal{O}(n \log(n))$, gdyż dopasowanie zielonej krzywej $a \cdot n \log(n)$ do danych jest znacznie lepsze. Trudność związana z teoretycznym wyznaczeniem złożoności czasowej tego etapu bierze się z silnej zależności pomiędzy strukturą grafu a liczbą wywołań poszczególnych funkcji (np. rekursja metody `add_node` opisana w poprzednim raporcie). Warto zwrócić uwagę na rosnące wraz z liczbą wierzchołków niepewności punktów pomiarowych. Jest to związane z tym, że liczba wierzchołków grafu w bardzo niewielkim stopniu mówi coś o grafie i generowane instancje, nawet tej samej wielkości, mogą się miedzy sobą bardzo różnić, zwłaszcza jeśli liczba wierzchołków jest rzędu kilku tysięcy.

5 Podział pracy

- **Filip Mieszkowski** - opracowanie oraz implementacja algorytmu rekonstrukcji, opracowanie oraz implementacja algorytmu generowania instancji. Opis algorytmu w głównej dokumentacji.
- **Stanisław Kurzątkowski** - opracowanie oraz implementacja algorytmu walidacji, eksperymenty w dokumentacji końcowej, dodatkowe elementy dokumentacji głównej.