# Przygotowanie do Olimpiady Informatycznej

### Co sprawdza Olimpiada?

Nie ma prostej reguły, dającej pewność osiągnięcia sukcesu w Olimpiadzie. Zazwyczaj finaliści wyróżniają się umiejętnością myślenia w sposób nieszablonowy, wymyślania nietrywialnych rozwiązań, bezbłędnością implementacji programów, pewną intuicją już podczas czytania treści zadań i umiejętnością wyszukiwania błędów w swoich rozwiązaniach.

Olimpiada sprawdza nie tylko umiejętności typowo informatyczne. Określony jest bowiem pewien kanon standardowych metod, algorytmów i struktur danych, których często można użyć w swoim rozwiązaniu. Znajomość algorytmów i struktur danych wraz z umiejętnością implementacji jest więc pierwszym z kroków, jakie musi uczynić każdy, który chce osiągnąć poważny sukces w Olimpiadzie. Na szczęście, sama wiedza choć jest bardzo potrzebna, nie jest wystarczająca. Żadne zadanie na Olimpiadzie nie polega na prostym zaimplementowaniu algorytmu znanego i wcześniej publikowanego. Zauważenie jak przekształcić dane, które otrzymujemy na wejściu, że jakieś dwa problemy są sobie równoważne, wreszcie, że możemy zastosować połączenie znanych technik jest najczęściej nietrywialne. To główna część każdego zadania olimpijskiego. Nie ma innej metody na wyćwiczenie umiejętności wymyślania rozwiązań, używania algorytmów we właściwym miejscu i wybór odpowiednich struktur danych, jak trening. Rozwiązanie dużej liczby zadań pozwala zdobyć doświadczenie, które bardzo przydaje się na zawodach. Do tego dochodzi bardzo ważna umiejętność — szybka i przede wszystkim poprawna implementacja rozwiązania, które się wymyśliło. Przy ocenie automatycznej, bardzo często zdarza się tak, że jeden zły znak, wśród ponad tysiąca w kodzie źródłowym zabiera zawodnikom nawet wszystkie punkty za dane zadanie, bo program z błędem nie jest w stanie rozwiązać nawet najprostszych testów.

Dlatego tak ważną, a często pomijaną umiejętnością, która decyduje o sukcesie na zawodach jest testowanie rozwiązań. Testy należy układać w odpowiedni sposób, aby sprawdzić każdy element rozwiązania. Do tego dochodzi element bardzo ważny, a moim zdaniem najważniejszy — strategia. W takim razie z całego tego akapitu może wynikać, że wszystko jest bardzo ważne. Jest to prawda. Schemat punktacji zbudowany jest tak, że wszystkie wyżej wymienione umiejętności muszą iść ze sobą w parze. Nie ma się jednak co martwić, wszystkich tych umiejętności uczy się w praktyce nie zdając sobie z tego sprawy. Podczas rozwiązywania zadań, często zdarzy nam się niespodziewane zero, długo będziemy wtedy szukać błędu — kiedy znajdziemy już błąd, na długo zapamiętamy ten błąd. Niestety, doświadczenia w startach, elementach strategicznych nie zdobędziemy „na sucho". Bardzo ważne jest zatem, żeby próbować swoich sił, nawet wtedy, kiedy obiektywnie patrząc nie ma się szans na zwycięstwo. Udział w drugim etapie lub w finale, możliwość poznania warunków, każdemu kto startuje drugi raz wychodzi na duży plus.

### Jak nauczyć się algorytmów i struktur danych?

Przede wszystkim jest potrzebna umiejętność swobodnego programowania w jednym z dozwolonych języków. Trudno ocenić, który z języków jest lepszy, łatwiejszy lub daje większe szanse na Olimpiadzie. Moim zdaniem, jeśli myśli się poważnie o Olimpiadzie, a dopiero się zaczyna — powinno się uczyć języka C++. Dlaczego? Regulamin dopuszcza stosowanie biblioteki STL, co w dużym stopniu odciąża implementacyjnie w części zadań. Na przykład: w zadaniach grafowych nie musimy implementować sami list, stos i kolejka także są już dostępne. Oprócz tego można użyć także o wiele bardziej skomplikowanych struktur: np. struktur słownikowych `set` oraz `map`. Ich efektywna implementacja samemu przekracza możliwości 99,9% zawodników. Pisanie tych struktur samemu podczas pięciogodzinnych sesji jest bardzo, bardzo ryzykowne,

zabiera sporo czasu, a miejsc na zrobienie błędu jest co niemiara. Do dyspozycji jest także bardzo szybka metoda sortowania — znowu nie trzeba implementować samemu. Sądzę zatem, że znajomość C++ w znacznym stopniu odciąża zawodnika od implementowania niektórych struktur. Oczywiście, nie twierdzę, że wygranie Olimpiady pisząc tylko w Pascalu czy C nie jest możliwe. Często jednak jest to nieco trudniejsze, w szczególności dla początkujących. Zanim bowiem będziemy mogli wniknąć w trudniejsze metody, będzie trzeba przejmować się mało istotnymi (z punktu widzenia programisty C++) błahostkami. Warto się więc zastanowić, czy nie dobrze byłoby skorzystać z dobrodziejstw STL, których możemy używać na Olimpiadzie. Nie trzeba znać wszystkich tajników języka, po prostu można zatrzymać się na takim poziomie, na którym stwierdzamy, że język programowania nie przeszkodzi nam w skutecznym zaimplementowaniu metody, którą wymyślimy. Odradzam zatem książki typu „Thinking in C++", czy „Symfonia C++". Zbyt wiele tam szczegółów, których znać nie trzeba, a postać książki jest zbyt mało skondensowana, by małym nakładem czasu zyskać pewną swobodę i wprawę. O wiele bardziej polecam kursy internetowe, np. na stronie Młodzieżowej Akademii Informatycznej (`www.main.edu.pl`). Jest tam dostępny kurs C++ oraz Pascala. Plusem kursu jest przygotowanie pod kątem Olimpiady i zadań typowo algorytmicznym oraz dostępność zadań na trenowanie metod, które dopiero się poznało.

### Jaką dobrać strategię?

Przejście przez pierwszy etap Olimpiady nie jest trudne. Progi kwalifikujące do drugiego etapu z ostatnich lat kształtowały się tak: 201, 138, 145, 100, 209 punktów. Łatwo zauważyć, że wykonanie dwóch zadań w pełni prawidłowo i dołożenie do tego rozwiązań poprawnych, ale nieoptymalnych (dalej potocznie zwanych „brutów") gwarantowało sukces w ostatnich latach (każde z pięciu zadań jest punktowane w skali 0–100). Olimpiada Informatyczna jest jednak nieprzewidywalna. Pełnej pewności nie ma nigdy. Biorąc pod uwagę niebezpieczeństwo popełnienia błędu w jednym lub więcej z rozwiązań, w miarę bezpieczna granica pojawia się, gdy jesteśmy pewni trzech swoich rozwiązań. Pomocą służą testy z forum Olimpiady. Należy pamiętać, że testów nigdy nie za wiele. Wielokrotnie zdarzały się sytuacje, w których nawet najtrudniejsze testy z forum nie wykrywały błędów w rozwiązaniach zawodników i bardzo wiele osób czekających na 100 punktów za dane zadanie kończyło nawet z zerem. Testy przykładowe, które wyświetla system po wysłaniu także nie są żadnym wskaźnikiem. Układane są bowiem tak, aby każdy błędny program (w miarę możliwości) je zaliczył. Są to testy najprostsze z możliwych. Wynik OK na tych testach absolutnie nie świadczy o poprawności rozwiązania, ani o uzyskaniu jakichkolwiek punktów. Należy więc podczas pierwszego etapu sumiennie rozwiązywać zadania i cały czas próbować powiększyć przewidywany dorobek punktowy. Zapasu nigdy za wiele. **Zdecydowana** większość początkujących zawodników jest **bardzo** niemile zaskoczona wynikiem oceny rozwiązań na prawdziwych, punktowanych testach. Należy więc być bardzo ostrożnym przy obliczaniu spodziewanej liczby punktów. Dobrze radzę wziąć sobie te zdania mocno do serca. Nie ma takiej liczby testów, liczby przetestowanych sposobów, które dają pewność i zwalniają z dalszego testowania — w szczególności na pierwszym etapie, kiedy jest dużo czasu.

Mimo wszystkich trudności, każdy kto włoży odpowiednio wiele pracy i dogłębnie przeanalizuje zadania, podczas miesiąca trwania pierwszego etapu, powinien być w stanie zdobyć wystarczającą liczbę punktów by przejść do drugiego etapu. Ten właśnie etap odsiewa prawie wszystkich niedoświadczonych zawodników. Progi kwalifikacyjne do finału z ostatnich lat kształtowały się następująco: 62, 124, 130, 102, 77 punktów. W obliczu czterech (najprawdopodobniej) zadań i maksymalnej liczby punktów wynoszącej 400 punktów, liczby te napawają optymizmem. Dlaczego więc tak wielu osobom (około 85%) przejście z drugiego etapu do finału się nie udaje? Wynika to po części z „brutalności" systemu oceny. Każdego dnia podczas dru-

giego etapu zawodów do rozwiązania są dwa zadania. Zazwyczaj (nie jest to regułą) — jedno z nich jest łatwiejsze, a drugie rozwiązują tylko najlepsi. Kluczowe jest rozpoznanie łatwiejszego zadania. Niestety, nie jest to łatwe. Na początku, gdy czyta się zadania wszystko zasłonięte jest historyjką, po chwili udaje się nam sformułować problem algorytmiczny ukryty w historyjce. Często jednak okazuje się, że mimo tego nie wiadomo, które zadanie jest łatwiejsze i na którym się skupić. Bardzo często okazuje się, że zawodnik nie opanował jeszcze wszystkich niezbędnych algorytmów i nie potrafi sobie na przykład poradzić z żadnym zadaniem grafowym. Tymczasem może właśnie to zadanie jest łatwiejsze? Wówczas marne szanse na zyskanie sensownego rezultatu danego dnia.

Na szczęście Olimpiada (jako tako) wybacza błędy na drugim etapie. Niezależnie od tego jak bardzo zepsuje się pierwszy dzień zawodów (nawet uzyska się 0 punktów), zawsze jest spora szansa na finał uzyskując dobry wynik dnia drugiego. W większości przypadków kluczem do zwycięstwa jest prawidłowe i efektywne rozwiązanie jednego zadania (uzyskanie za jedno zadanie 100 punktów). Dla pewności należy do tego dołożyć jeszcze kilka brutów lub lepiej drugie rozwiązanie poprawne. Niestety, mówi się łatwo, a wykonać trudniej. Wymyślenie poprawnego i szybkiego rozwiązania na drugim etapie jest trudne, jest mało czasu, a wiadomo, że jest przecież jeszcze drugie nietknięte zadanie, nawet jak się już wymyśli, skąd wiadomo, że się nie pomyliło, może trzeba dłużej testować, może lepiej po dwóch godzinach bezskutecznego myślenia próbować rozwiązywać drugie zadanie? Przed takimi dylematami stoi każdy zawodnik na Olimpiadzie.

Trudno mi opisać poprawną strategię na jeden dzień zawodów. Dużo zależy od tego co chcemy osiągnąć na drugim etapie. Czasami zależy nam na maksymalizacji wyniku oczekiwanego, który uzyskamy, jeśli wszystko pójdzie dobrze, czasami maksymalizujemy prawdopodobieństwo kwalifikacji do finału, czasami chcemy mieć po prostu największą liczbę poprawnych rozwiązań. Przeciętnemu uczestnikowi drugiego etapu chodzi jednak o dostanie się do finału — na tym się skupię.

Pierwszy dzień zawodów przynosi zawsze sporo nerwów. Dobry wynik na pierwszym dniu — wymyślenie wzorcówki (rozwiązania na 100 punktów), lub chociażby uratowanie punktów brutami przynosi ogromną ulgę drugiego dnia i zasadniczo zmienia strategię na drugi dzień zawodów. Naszym celem powinno być przede wszystkim doprowadzenie jednego zadania do poprawności i efektywności. Mając pięć godzin czasu warto nawet jedną poświęcić na wybór zadania, które się rozwiązuje. Ciągnięcie dwóch zadań na raz w przypadku, gdy jest się dopiero amatorem w najlepszym wypadku (oczywiście pomijam wyjątki) kończy się napisaniem dwóch poprawnych, ale nieefektywnych rozwiązań. Jeśli się to uda można liczyć na w miarę bezpieczne 50–60 punktów. Nie uwalnia nas to niestety od wymyślenia wzorcówki dnia drugiego. Nie jest to więc to o co powinno chodzić przeciętnemu uczestnikowi Olimpiady. Celem numer jeden jest więc wymyślenie rozwiązania wzorcowego do jednego z zadań. Kiedy to się w końcu udaje — nie jest to jeszcze sukces. W pełni poprawne rozwiązanie w głowie jest nic nie warte, póki nie zaimplementujemy go poprawnie. Tutaj od razu przestrzegam przed natychmiastowym klepaniem w klawiaturę. Naprawdę warto, póki jest czas, poświęcić jeszcze chwilę na dokładne przemyślenie jak zaimplementować to, co się wymyśliło. Wbrew pozorom oszczędza to bardzo wiele czasu. *Myśl dwa razy, abyś nie musiał kodzić trzy razy.*

Dobrze — wymyśliliśmy rozwiązanie, wiemy jak je napisać, pora zacząć pisać. Najgorsze co może się okazać to jeśli podczas pisania algorytmu stwierdzimy, że jednak nie działa lub trzeba go w jakiś sposób przemodelować. Świadczy to tylko o tym, że rozwiązanie nie zostało dokładnie przemyślane. Należy jeszcze raz wrócić do kartki (jest to często bolesne — chwilę wcześniej bowiem myśleliśmy, że już mamy jedno zadanie). Kiedy jednak udało się w końcu zaimplementować rozwiązanie, należy je bardzo dokładnie przetestować. Bardzo często na początku kod zawiera jeszcze w sobie kilka błędów. Test przykładowy na pewno nie jest od tego, żeby te błędy wy-

chwytywać. Należy ułożyć własne testy. Muszą one sprawdzać jak najwięcej i być maksymalnie chamskie dla danego rozwiązania. Wyobraźmy sobie, że chcemy obalić to rozwiązanie. Kiedy nam się po usilnych próbach nie udaje możemy albo uznać, że jest to już poprawne rozwiązanie (ryzykowne), albo zastanowić się nad zaimplementowaniem bruta (o ile to możliwe). Można wtedy dopisać jeszcze generator dużych testów i porównać odpowiedzi na dużej liczbie testów. Warto też spróbować wygenerować testy bardziej skomplikowane niż losowe — często można to zrobić za pomocą prostych schematów pozwalających nawet na ręczne wyliczenie odpowiedzi. Pamiętajmy, testów nigdy za wiele. Organizatorzy układają testy najtrudniejsze, jakie tylko potrafią. Programy zawodników przetwarzają miliony liczb, znaków, wierzchołków, punktów podczas uruchomienia. Kilka testów na kartce ma nas upewnić o poprawności programu, który za chwilę będzie „rzucany w taką przepaść"?

Ważna uwaga. Przy obliczaniu odpowiedzi do wymyślonych lub wygenerowanych testów nie korzystajmy z algorytmu, który wymyśliliśmy. Może się przecież okazać, że jest on niepoprawny, nie daje optymalnej odpowiedzi, jest tylko jakąś heurystyką. Kiedy testujemy nasze rozwiązanie już zaimplementowane, musimy ze wszystkich sił spróbować zapomnieć o tym jak nasz program wylicza odpowiedź i skupić się na wymyśleniu jak najtrudniejszych testów oraz ręcznym policzeniu dla nich prawidłowej odpowiedzi. Dobrym pomysłem przygotować kilka testów najpierw na kartce, a dopiero potem testować. Może się bowiem okazać, że wykryjemy jakiś błąd w algorytmie/implementacji i uznamy to za błahostkę łatwą do poprawienia. Być może następne testy, lub testy, które nasz program przeszedł pomyślnie uświadomią nam, że jednak sprawa nie jest taka prosta. Nie ukrywam, że po „zaklepaniu" kodu, znajdowanie błędów w algorytmie jest okropne. Wskazuje to na niedokładne przemyślenie i zbyt szybkie przejście do fazy implementacji. Podczas pięciu godzin zdarzy nam się na pewno nie raz popełnić jakiś błąd strategiczny. Trzeba jednak umieć sobie z nimi radzić.

Co jeśli mieliśmy rozwiązanie, które później obaliliśmy lub nie udało nam się nic wymyślić, a czasu jest coraz mniej? Być może wtedy warto przełączyć się na plan B. Jeśli da się wymyślić proste, poprawne i nieoptymalne rozwiązania — warto je zaimplementować. Podczas pierwszego dnia zawodów można wywalczyć sobie prawie pewność, że jest się w finale (pisząc wzorcówkę i bruta lub lepiej — dwie wzorcówki, co jest jednak mało realne tak od razu skoro czytasz ten tekst), można wywalczyć sobie praktycznie prostą drogę do finału (pisząc wzorcówkę), można wywalczyć sobie duże szanse na finał (piszac poprawne dwa bruty — o ile to w danym dniu wykonalne) lub można zdobyć po prostu jakieś punkty, stanowiące jakiś ułamek progu. Zero jest tutaj wynikiem najgorszym — dającym wciąż spore szanse na finał, ale szanse te muszą być poparte bardzo dobrym wynikiem dnia drugiego. Róbmy co się da, aby ulżyć sobie na drugim dniu zawodów.

Strategia na dzień drugi w dużej mierze zależy od naszego raportu i rzecz jasna raportów przeciwników. Jeśli wypadliśmy bardzo dobrze (najczęściej oznacza to wynik powyżej 100 punktów pierwszego dnia) mamy już z górki. Moim zdaniem najlepsza strategia na finał to wówczas czytanie zadań, skoncentrowanie się na napisaniu dwóch poprawnych, niekoniecznie szybkich i efektywnych rozwiązań. Zależy nam po prostu na pewnych punktach. Pisanie wzorcówki jest wtedy większym ryzykiem — jeśli się pomylimy, prawdopodobnie dostaniemy 0. Oczywiście istnieje ryzyko, że nasza niezła suma punktów nie wystarczy na finał. Olimpiada jest nieprzewidywalna. Należy się z tym liczyć. Mimo to, uzyskanie wyniku w granicach 200 punktów za oba dni jest już bardzo dobrym wynikiem dającym bardzo duże szanse na przejście do trzeciego etapu. Po napisaniu brutów zawsze przecież można próbować myśleć nad wzorcówką do choćby jednego z zadań. Mało tego, mamy już bruta, z którym możemy przetestować nasze rozwiązanie. Dodatkowo, jeśli nie mamy pewności możemy hybrydować bruta i niepewną wzorcówkę: dla małych danych uruchomić bruta, a dla większych niepewny kod. Jak widać, dobry wynik na pierwszym dniu daje nam bardzo wiele alternatyw i przede wszystkim mniejsze wymagania

co do zdobyczy punktowej. Gorzej sprawy się mają jeśli nasz wynik z pierwszego dnia był mizerny. Wtedy celem jest po prostu zaimplementowanie rozwiązania wzorcowego. Należy wtedy nawet podjąć duże ryzyko. Nie ma już czego ratować (zdarzają się co prawda progi niższe niż 100 punktów, ale z reguły bez wzorcówki jest bardzo ciężko przejść do finału). Nadal jednak kluczowe jest właściwe wybranie zadania. Dodatkowo, w przypadku zera za pierwszy dzień, nawet 100 punktów za drugi może nie wystarczyć. Wtedy trzeba jeszcze próbować zdobyć jakieś punkty na drugim zadaniu. Jak widać: sytuacja na drugim dniu w dużej mierze zależy od wyniku pierwszego dnia.

Strategię na finał trudno w ogóle opisywać, gdyż jest jeszcze więcej czynników: między innymi liczba zadań, ich trudność, którą w ogóle trudno oszacować, poziom finalistów, typ zadań. Wierzę, że Ci, którzy dostali się na finał, wiedzą jak sobie już dalej radzić. Życzę tego wszystkim czytelnikom.

### I co? Już?

Strategie i pomysły na to jak znaleźć się w finale i osiągnąć tam dobry wynik, które podałem powyżej nie są jedynymi słusznymi. Nikt nie może dać gwarancji sukcesu, mimo to ufam, że rady i sposoby, które tu przedstawiłem, warte są chociażby wspomnienia. Mam nadzieję, że chociaż komuś te rady pomogą, a wyniki na drugim etapie i finale będą satysfakcjonujące.

*Autor: Karol Pokorski*