



Zadania z biblioteczkami considered harmful

2019-04-11

| | |
|---|---|
|  | Moje poglądy trochę się zmieniły od kiedy to napisałem; stwierdziłem że w praktyce okazjonalna dyskwalifikacja jak ktoś zrobi coś nie tak jest mniejszym nakładem pracy niż przeprojektowanie całego systemu żeby był odporny (zwłaszcza, że organizatorzy i tak muszą przejrzeć wszystkie zgłoszenia). Poza tym można napisać własną wersję biblioteczki która i tak wszystko wczytuje i wypisuje na standardowe strumienie. Więc chociaż biblioteczki nadal są w mojej opinii trochę gorszym rozwiązaniem, nie są aż tak złe żebym nadal był inkwizytorem alternatywnych rozwiązań. |
|  | Powszechnie wiadomo, że Olimpiada Informatyczna, ściągając formułę konkursu od LOGII, nieświadomie zepsuła tradycyjną sztukę mierzenia talentu programistycznego na kilka ważnych sposobów. Są to oczywistości takie jak absurdalnie małe limity czasowe (żeby przyszczędzić na czasie serwera przy sprawdzaniu rozwiązań), ale są to też rzeczy, o których niektórzy mogą nawet nigdy krytycznie nie pomyśleć — mam na myśli zadania z biblioteczkami. |
| | Ponieważ <i>każde</i> warsztaty i <i>każdy</i> obóz informatyczny jaki znam przygotowują ostatecznie do Olimpiady, gdzie mogą wystąpić takie zadania, <i>wszędzie</i> są takie zadania, "dla naszego własnego dobra, żebyśmy byli przygotowani". |

Łatwo je zaakceptować jako coś naturalnego, nauczyć się ich używać, pogodzić się ze wszystkimi wadami i mieć po prostu nadzieję, że będą się pojawiać jak najrzadziej. To jest jednak dokładnie podejście, jakiego oczekują od nas ludzie mający sekretny interes w utrzymywaniu tej szkodliwej tradycji — naszym obowiązkiem wobec przyszłych pokoleń jest się im sprzeciwić.

Zrozumieć

Sam koncept biblioteczek wydaje się prosty, intuicyjny i genialny: dzięki temu, że program musi wywoływać nasze funkcje, może tylko na bieżąco odpowiadać na zapytania, więc można przyznawać punkty tylko algorytmom online! Hurra! Kolejny problem, który występowałby rzadziej gdybyśmy nie mieli na niego rozwiązania, rozwiązany!

Jednak:

Wydłużanie treści zadania

Chociaż dodanie jednego pliku do kompilacji może wydawać się proste dla doświadczonych osób, nie wszystkie osoby są doświadczone. Aby nie ustawiać zawodów przeciwko nim, w każdym zadaniu musi być wyjaśnione: skąd wziąć przykładową biblioteczkę, gdzie jest przykładowe błędne rozwiązanie, jak to skompilować, jak to testować, jakie są dokładne typy deklaracji funkcji.

W ten sposób na OI 30-70% treści każdego zadania z biblioteczką to nudne, czysto techniczne wyjaśnienia niezwiązane z treścią zadania ani algorytmiką.

Narzucanie konwencji

W zadaniach muszą być podane dokładne interfejsy komunikacji z rozwiązaniem. Jedynym osiągnięciem wymuszania użycia w jednym zadaniu `long long WartośćNaPozycji()` a w innym `int value_at_pos()` jest zmniejszenie czytelności i walorów estetycznych kodu.

Bezpieczeństwo

Możemy chyba się zgodzić, że bezpieczeństwo nie jest mocną stroną języka C++. Korupcja stosu to jedna z podstawowych operacji. Konstruktory obiektów klas globalnych wywołują się przed funkcją `main()`, więc naiwne zablokowanie tej funkcji w biblioteczce w niczym nie pomaga.


Python z kolei nawet nie ukrywa, że wszystko jest możliwe z odpowiednim modulem. Funkcja `globals()` zwracająca wszystkie funkcje globalne jest wbudowana. Moduł `ast` służy do wygodnej modyfikacji kodu w czasie wykonywania.

Jedynie co chroni kontesty przed masowym hakowaniem sprawdzarek w taki sposób to utrzymywanie ich kodów źródłowych w tajemnicy. Nazywamy to *security by obscurity* i jest to niebezpieczny stan rzeczy: jeden wyciek kodu źródłowego i wszystkie podobnie zaimplementowane sprawdzarki nagle są podatne na ataki.

Sprawia to oczywiście, że nie znam ani jednego przypadku gdy sprawdzarka została publicznie udostępniona, nawet gdy same pliki z testami były.

Na XXVI Olimpiadzie Informatycznej w zadaniu oczekującym, że rozwiązanie wywoła `give_answer(n)`, pewien uczestnik w Pythonie zwrócił mniej-więcej taki obiekt:

```
class X(int):
    def __eq__(self, other):
        return other != 0
    def __int__(self):
        return self
give_answer(X())
```



Dla osób niepełnych w Pythonie: porównanie obiektu klasy X z dowolną liczbą nierówną zero zwraca wartość prawdziwą. Dodatkowo, próba zamiany takiego obiektu na zwykłą liczbę całkowitą używając funkcji `int()` jest bezskuteczna (w Pythonie 3.4 używanym wtedy na olimpiadzie — zostało to naprawione w nowszych wersjach).

Jak zauważył pewien anonimowy algorytmik, jest to jedyny udokumentowany przypadek wykorzystania polimorfizmu na Olimpiadzie Informatycznej.

Prawdopodobnie jednak większość biblioteczek jest zabezpieczona tak, że program sprawdzający wypisuje jakiś hasz wejścia oraz odpowiedzi zwrócone przez program, co znacznie utrudnia hakowanie.

Oczywiście, takie ataki przestają działać w momencie gdy ktoś zajrzy do kodu, czyli z personalnego doświadczenia przed 20:00 tego samego dnia. Większość osób to wie i nawet nie próbuje marnować swojego czasu.

Można też założyć, że większość ludzi jest uczciwa i nie wykorzysta żadnej oczywistej dziury w systemie w celu zdobycia przewagi. Taki sposób myślenia sprawia tylko, że będziemy zaskoczeni gdy nieuchronnie się znajdzie wyjątek.

Lepsza alternatywa

Więc, mając nadzieję, że przekonałem wszystkich zwolenników starego systemu, przedstawię teraz alternatywę. Nie jest to całkowita likwidacja zadań domagających się rozwiązań online — to bardzo ciekawa grupa algorytmów.

Wszyscy wiemy, że wejście i wyjście zarówno w Pythonie jak i C++ jest *asynchroniczne*. I to jest właśnie problemem gdy chcemy algorytm działający online — nic nie powstrzymuje rozwiązania przed wczytaniem wszystkich zapytań i odpowiedziem na wszystkie naraz.

Hmm... jak można rozwiązać ten problem?

“*Wiem: sprawię że wszyscy się będą męczyć z dodatkowymi nagłówkami i ich kompilacją, narzucę im mój styl nazywania funkcji, a do tego wszystkiego nawet nie załatam oczywistych dziur w bezpieczeństwie i po prostu cicho zdyskwalifikuję każdego kto jakąś znajdzie!*

— Ktoś najwyraźniej wpływowy
dzień przed wprowadzeniem zadań z biblioteczkami

Nie.

Otóż rozwiązaniem jest *synchroniczne* wejście i wyjście. Po prostu kazać rozwiązaniu uczestnika flushować odpowiedź po każdym zapytaniu i nie przysyłać kolejnego wiersza wejścia jeżeli nie dostanie się odpowiedzi. Wszystko przez bezpieczny, znany i przyjazny strumień tekstowy, niemożliwy do zhackowania.

Okazuje się, że jest to system od dawna szczęśliwie używany przez Codeforces.