Wykład o strategii (Tomasz Nowak)

- Przypomnienie zasad drugiego etapu OI
  - pytania tylko w pierwszych dwóch godzinach, odpowiedzi tak / nie / niepoprawne pytanie / odpowiedź wynika z treści zadania / bez odpowiedzi. Pytania techniczne przez całe zawody
  - często wchodzić na pytania (o tym jak Michał nie zauważył zmiany podzadania na BOI)
  - drukowanie i kopia zapasowa, skrypt submit (o tym, jak chyba dwa lata temu przesunięto o godzinę drugi etap, bo w jednym mieście padło zasilanie kompów z live cd)
  - max 15 submitów, liczy się ostatni submit
  - w plikach są przykładowe i limity czasu
  - dokumentacja
- Minimum, by się dostać do finału
  - ostatnie progi: 200, 162, 200, 87, 80, 168, 140, 206, 175, 62. Łatwo wyciągnąć wnioski
  - w pierwszym dniu oczywiście, najlepiej celować w wzorcówkę i bruta albo dwa bruty, nie da się niczego mądrego powiedzieć
  - zależnie od wyniku w pierwszym dniu, strategia się zmienia w drugim dniu (sytuacja wygląda tak, że powinno się sklepać wzorcówkę, albo można na spokojnie bezpieczniej dwa bruty)
  - bardzo kluczowe jest rozpoznanie łatwiejszego zadania, które zazwyczaj jest (Justynka na drugim etapie OIG (nie OI) wyzerowała bo klepała 2h hardkora)
  - nieprzewidywalność i niepewność i brak wiedzy
- Idealny „early game“ podczas contestu
  - *zrozumieć* *oba* zadania podczas pierwsze pół godziny, albo nawet godziny. Nigdy nie kminić tylko nad jednym zadaniem. Nigdy nie dopuścić do sytuacji, gdzie się klepie zadanko, a drugie jest jeszcze nieprzeczytane (jakieś przykłady takich błędów z życia)
  - zrobić wiele swoich przykładów, zamiast przykładowych z treści (podchwytliwe treści i przykładowe)
  - potwierdzić, że się rozumie treść, na podstawie przykładowych
  - nigdy nie zaszkodzi przeczytać treść jeszcze raz, od początku do końca (pierwszy dzień warsztatów staszicowych)
  - zrozumieć każdy szczegół treści, szczególnie dziwne warunki (po co one są albo co ułatwiają)
- Doświadczenie w rozwiązywaniu zadań
  - słabą intuicję można nadrobić dużym doświadczeniem
  - rozwiązanie może korzystać z techniki lub obserwacji, którą znamy
  - intuicja - czujemy z jakiego podejścia/przekształcenia należy skorzystać
  - doświadczenie - mamy dużą listę pomysłów, z których można skorzystać w danym zadaniu
  - dużo osób ma spore doświadczenie, ale nie zdaje sobie z tego sprawy (jak je wykorzystać?)
  - może warto przed OIem przejrzeć przez zadania, które się wbiło, i przypomnieć sobie rozwiązania i wyciągnąć z nich całą „esencję”? Dla każdego zadania skrócić jak najbardziej ideę rozwiązania, w taki sposób, że gdyby się sobie powiedziało o tym, kiedy się kminiło zadanko, to to by znacząco pomogło
  - ostatnio zadania są częściej na techniki, niż obserwacje. Oznacza to większy nacisk na doświadczenie
  - przykład: znajomość problemów NP-trudnych
- Lista technik to złoto
  - przedstawić kilka zadanek na każde podejście
  - dlaczego lista technik może czasem znacząco pomóc w rozwiązaniu zadania

  - podkreślić, że lista technik to złoto
  - nakrzyczeć, by znali na pamięć podejścia, gdyż może to pomóc
  - wyjaśnić, że nie ma powodu się jej wstydzić
- Pewność siebie - dlaczego czasem się (niepotrzebnie) poddajemy
  - Szczecin to zadupie, nie mamy osób, z którymi możemy się porównywać
  - jesteśmy lepsi, niż nam się wydaje
  - moja historyjka na finale OI - wbiłem jedno zadanko, pomyślałem „mm, ładne i trudne zadanie, po to tu przyjechałem. Ok, jestem już zadowolony ze swojego wyniku". Co z tego wyszło? Nie wbiłem łatwych, a na pewno mnie na to było stać. Patrzyłem na te zadania i wgl nie miałem podejścia by je wbić na 100.
  - często się zdarza, że zatrzymujemy się przy łatwiejszym zadanku i się poddajemy, a jesteśmy w stanie zrobić nawet trudniejsze. „Flow"
  - pewność siebie może mieć wpływ na wynik na conteście
  - jak budować pewność siebie? Porobić / przypomnieć sobie zadanka drugoetapowe
  - nie daj się przerazić treści, często autorzy specjalnie próbują przerazić
  - nie bój się algorytmu, tylko go sklep
  - celować wyżej
  - załóż, że zadania są łatwe
- Co zrobić, gdy ma się laga mózgu?
  - opcja 1: nie walczyć z tym, tylko przeczekać. Zrobić sobie przerwę, pójść zjeść kanapkę, pójść do toalety
  - opcja 2: spróbować „zresetować" kminę, tzn spojrzeć na zadanie od nowa, przypomnieć sobie jakie obserwacje zostały zrobione (albo jakie obserwacje by się przydały), przypomnieć sobie w głowie listę technik (szczególnie listę podejść), spróbować z nich maksymalnie skorzystać, spróbować przypomnieć sobie podobne zadanka, które się robiło, albo podobne podejścia. Powołać się na swoje doświadczenie i wgl
- Dobry „midgame" podczas contestu
  - klepać najpierw bruty (nie zawsze ale zazwyczaj tak) - dlaczego to jest bardzo bardzo bardzo dobre (pomaga przy dużej liczbie kwestii i sytuacji), mamy już na starcie jakieś punkciki
  - dobre jest klepać kolejne podzadania, jednak czasem podzadania to bait
  - nie zapominać, że są dwa zadania, oraz nie wiemy które jest łatwiejsze
  - często zdarza mi się, że nic nie mam podczas pierwszych trzech godzin contestu, a potem maxuję (narysować wykres „ile mi się wydaje że mam" oraz „ile w praktyce wymyśliłem")
  - rozwiązywanie zadań często u mnie wygląda tak, że próbuję wyczerpać pomysły/podejścia/techniki które mi się przyjdą do głowy z doświadczenia, albo szukam obserwacji (przy czym wiem jakie mniej więcej szukać), zazwyczaj to się udaje
  - nie stresować się xd (wiem że łatwo powiedzieć), naprawdę nie jest koniec świata jak się obudzimy w połowie i pomyśli się „ej, ale ja jeszcze nie mam, shit, no to koniec"
- Jak nie marnować czasu?
  - Justyna w pierwszej klasie na drugim etapie OI
  - *Jak macie geo to uciekajcie* - Kacper Walentynowicz
  - W momencie, kiedy myślę, że powinienem klepać od nowa, jak tego nie zrobię to zawsze żałuję
  - *Czasami się opłaca klepać od nowa, zanim to będzie za późno* - Justyna Jaworska
  - Klepać rozwiązania etapowo (kolejne podzadania), dać przykłady z pisania przedział-przedział. Sprawdzać kolejne części kodu. Jak się skończy fragment kodu, powinno się zastanowić czy na pewno jest git i czy można go przetestować, bo potem marnuje się czas na debuggowanie wszystkiego naraz (binsearch wielu rzeczy naraz po kodzie jest trudniejszy niż zwykły binsearch)

  - spróbować przekminić kod, zanim się spróbuje go napisać (nie jest to dla każdego, ale spróbować nie zaszkodzi)
  - wkopanie się może być z trzech powodów: zła implementacja albo zostawienie jednego zadania albo klepanie heury
  - sztuczki implementacyjne: add i mul dla modulo, spróbować sprowadzić klepę do użycia znanego już rozwiązania, np. konstrukcja grafu i bfs
  - nagłówki: rozsądna opcja podczas contestu. Można np napisać wypisywacz vectora intów, a jak potrzeba to rozszerzyć do template<class T>
  - jak się jest w pewnym momencie niepewny czego się robi, to jest źle
  - poddanie się boli, ale może się opłacić
  - koksy kminią na zmianę nad zadaniami (przykład: tourist, ja, Michał)
- Dobry „endgame" podczas contestu
  - *Myśl dwa razy, abyś nie musiał kodzić trzy razy* - Karol Pokorski
  - *Jak widzę, że klepię gówno, to po prostu usuwam kod* - Marek Skiba
  - jak *bardzo* ważne jest testowanie rozwiązań (Michał zerował część P w staszicowych contestach)
  - można skorzystać z brutów sklepanych wcześniej, jednak bruty powinny nie mieć wspólnej części kodu z wzorcówką
  - generować wszystkie możliwe testy, shuffle'ować też jeżeli to może mieć znaczenie
  - poza testowaniem, co można zrobić? Limity czasu oraz limity pamięci oraz overflow. Jak je sprawdzić?
  - flagi kompilacji oraz narzędzie gdb (oraz oiejq OIa jeżeli jest)
  - TODO rozszerzyć o klepaniu, konieczności testowania, błędach przy tym
  - jak jest wiele testów w pliku, to najlepiej generować $t = 2$
  - na ich kompach, które są beznadziejne, najlepiej jest robić mało testów z dużym t zamiast dużo z małym t (nawet jak trzeba u siebie dodać istnienie t)
  - często opłaca się bardziej zasubmitować bruta, niż nieprzetestowaną wzorcówkę (zależy od dużo rzeczy)
- Co zapamiętać na pamięć przed OIem?
  - listę technik, szczególnie podejścia
  - flagi kompilacji
  - sposób użycja gdb
  - prostą wersję nagłówków (najlepiej swoją wersję)
  - w jaki sposób sprawdzić czas i pamięć programu
  - generatorka i sprawdzarka
- Jak dużo wyciągnąć z tego obozu
  - trzy OIowe zadanka bez odsłonięć, polecam pisać bruty. Czyli maksować EV wyniku zamiast nie patrzeć na prawdopodobieństwo zbugowania i celować w maxy
  - spróbować wyciągnąć z zadań jak najwięcej esencji
  - stały jest bardzo ważny, buduje praktykę i doświadczenie. Zazwyczaj nie pamięta się technik, których się nigdy nie klepało (zresztą to może powodować wkopienie się, tak jak Justynka w pierwszej klasie)
  - warsztaty technikowe
  - ponownie, stały jest bardzo ważny