Umijeće rješavanja zadataka

Igor Čanadi 2008.

Savjet 1

- Tijekom rješavanja zadatka probajte "imaginarnom kolegi" objasniti što se traži od vas u zadatku i što ste do sad zaključili.

Loš pristup rješavanju zadatka

- Pročitaš zadatak, shvatiš otprilike što trebaš napraviti
- Iskodiraš učitavanje
- Imaš neku ideju, iskodiraš ju nabrzaka
- Nakon dugo debagiranja shvatiš da je ideja krivo
- Popravljaš ideju, krpaš, nakon nekog vremena nema šanse da se snađeš u kodu

Koraci rješavanja zadatka

1. Čitanje zadatka
2. Smišljanje rješenja
3. Implementacija rješenja
4. Testiranje rješenja

1. Čitanje zadatka

- Pročitajte cijeli tekst jednom do dvaput
- Podcrtajte najvažnije
- Proučite ograničenja
- Proučite svojstva i povežite ih

Ograničenja

- Kolika bi trebala biti složenost?
  - O(n^2) -> DP
  - O(n log n) -> sortiranje, binary search
  - O(n) -> "direktan" pristup
- Smanjenje ograničenja
  - sažimanje
- IOI 2007 - training: "iz svakog grada izlazi najviše 10 cesta." -> 2^10 je dovoljno malo!

Svojstva

- Koja svojstva mi garantira tekst zadatka?
  - Jedinstveni brojevi
  - Točno jedan put između dva čvora -> stablo
  - Nema puta povratka -> DAG
  - Crni i bijeli čvorovi -> bipartitan graf
- Sva svojstva u tekstu zadatka podcrtajte

2. Smišljanje rješenja

- Kako mogu riješiti zadatak?
  - Prikazati kao graf?
  - Dinamičko programiranje?
  - Pametna struktura?
  - Upotrijebiti svojstva (ad-hoc)?
  - Brute force?
  - Divide and conquer?
  - Binary search?
  - Greedy?

Podjela na podprobleme

- Mogu li zadatak podijeliti na više podproblema koje mogu nezavisno riješiti?
- Smanjenje dimenzije

Dokazivanje točnosti

- Izvedite dokaz rješenja ako možete
- Objasnite rješenje "imaginarnom kolegi" i uvjerite ga u točnost rješenja
- Nemojte žuriti – 5 minuta više na dokaz rješenja može vam uštedjeti 1 sat implementacije pogrešnog rješenja
- Vratite se na podcrtane stvari iz teksta zadatka! Zadovoljava li vaše rješenje zadatak u potpunosti?

3. Implementacija rješenja

- Priprema terena
- Kodiranje
- Provjera

Priprema terena

- Pseudokod
  - Mora biti takav da netko samo na temelju teksta zadatka i pseudokoda može riješiti zadatak
  - Sređuje misli i znatno ubrzava vrijeme kodiranja

Priprema terena

- Dinamičko programiranje
  - Memoizacija ili ne?
  - Što moram pamtiti?
  - Koja manja rješenja moraju biti obrađena za računanje većeg rješenja? Orjentacije petlji?
  - Rekonstrukcija?
  - Memorijsko ograničenje!

Priprema terena

- Grafovi
  - Reprezentacija grafa u memoriji
- Brute force
  - Pruning? Kad mogu "odrezati granu"?
  - Običan backtracking, iterative deepening ili A*?

Priprema terena

- Analiza vremena izvršavanja
  - Hoće li se izvršiti u vremenu?
  - Koji je worst case?
  - Asimptotsko ubrzanje
    - Brže strukture
    - Precalculate
  - Smanjenje konstante
    - Najčešće se ne isplati

Kodiranje

- Na temelju pseudokoda pažljivo iskodiramo rješenje
- Lijep kod olakšava traženje grešaka
- Komentari pomažu

Pravo rješenje + Brute Force

- Ako niste 100% sigurni u vaše rješenje, dobra ideja je staviti uvjet na početak programa: ako je N dovoljno mali, pokreni brute force, inače pokreni pravo rješenje
- Brute force rješenje ionako morate implementirati za testiranje rješenja
- Ako neznate rješiti zadatak: Brute Force + heuristika

Provjera

- Nakon što se kod uspije kompajlirati, bacite još jednom oko jeste li implementirali sve što ste trebali i zadovoljili sve uvjete zadatka
- Pazite na veličinu alociranih polja – česta greška

4. Testiranje rješenja

- Sjetite se koliko puta ste zadatak slali na ACM i dobili Wrong Answer
- Šteta je izgubiti bodove na zadatku kojeg znaš rješiti zbog glupe greške
- Pristupite testiranju sa stavom da greška negdje postoji i da ju morate naći

Alati za testiranje

- Generator test primjera
- Brute Force rješenje
- Skripta koja automatizira proces

Što testirati?

- Granični slučajevi
- Mali random test primjeri
  - Testiraju točnost pomoću Brute Force rješenja
- Ogromni test primjeri
  - Testiraju time limit, SIGSEGV...
- Testirati sve slučajeve (npr. sva 4 smjera u zadatku s gridom)

Ne radi! Što sad?

- Prođite kroz kod još jednom – najlakše i najbrži način za pronalazak bugova
- Probajte pronaći najmanji test primjer na kojem vaše rješenje ne radi
- Ispisivanje međurezultata, ručna provjera

Zaključak

- Rješavanje zadataka zahtjeva sistematičan pristup i sređenost misli u svakom trenutku