Lab06 ariketak: Vertex_Cover bilaketa zuhaitzekin

1. **Vertex_cover posible bat balioztatu**

   Ariketa honetan `lab06_vertex_cover_2.py` fitxategia daukazu abiapuntu bezala, bertan `partial_validity_check` funtzioa inplementatu behar duzularik. Funtzio horretan, honakoa emanda:

   - 0ko, 1eko eta `None`-ez osatutako zerrenda bat, cover-aren nodoak adierazten dituena: i 1eko baten posizioa bada, orduan i. nodoa cover-ean dago. i 0ko baten posizioa bada, orduan i. nodoa ez dago cover-ean. i posizioa `None` batena bada, i. nodoa cover-ean dagoen ala ez ez dakigu.
   - eta ez zuzendutako grafo bat,

   erabaki cover-ean dauden nodoek grafoko ertz guztiak estaltzen dituzten edo ez.

   Funtzioa probatzeko `test()` metodoan dauden `partial_validity_check`-ri dagozkion `assert` eragiketa guztietan iruzkinak kendu behar dituzu.

2. **Bilaketa zuhaitz bitar bat erabilita vertex_cover bat topatu**

   `lab06_vertex_cover_2.py` fitxategian `vertex_cover_ tree` funtzioa topatuko duzu. Grafo man emanda, aldagaiak hasieratu eta `recursive_ vertex_cover` funtzioari deitzen dio, partzialki inplementatu behar duzuna.

   `recursive_vertex_cover` funtzioak vertex_cover minimo bat topatzen du, prozesatu ez den v nodo bat aukeratzen eta ondoren bilaketa zuhaitz baten bi adarrak sortuta. Adar bat v cover-ean sartzeari dagokio eta bestea cover-ean ez sartzeari.

   Inplementatu behar duzun zatiak oraindik baliozko cover bat eraikitzea dagoela frogatu beharra dauka. Ezinezkoa bada, `[1]*len(cover)` bueltatu behar du. Oraindik posiblea bada, oraindik prozesatu ez den v nodo bat topatu behar da. Jada cover-a osatuta egoteagatik v existitzen ez bada, cover-a bueltatu behar du. Bestela, v aukeratu eta funtzioan jada idatzita dagoen kodearen jarraitu. Zati hau ezin da aldatu.

   Funtzioa probatzeko `test()` metodoan dauden `vertex_cover_tree`-ri dagozkion `assert` eragiketa guztietan iruzkinak kendu behar dituzu.

3. **Bilaketa zuhaitz hirutar bat erabilita vertex_cover bat topatu**

   `lab06_vertex_cover_3.py` fitxategia ireki. Fitxategi hau aurreko ariketarenaren antzekoa da, `recursive_ vertex_cover`-ek bilaketa zuhaitzean hiru adar sortzen dituela kenduta.

Funtzio errekurtsibo honek vertex_cover minimo bat topatzen du, oraindik prozesatu ez diren bi u eta v nodo aukeratuz eta ondoren bilaketa zuhaitzean hiru adar sortuz. Lehenengo adarra cover-ean u sartu eta v ez sartzeari dagokio; bigarrena u ez sartu eta v sartzeari dagokio; eta hirugarrena bai u eta bai v sartzeari.

Inplementatu behar duzun zatiak oraindik baliozko cover bat eraikitzea dagoela frogatu beharra dauka. Ezinezkoa bada, `[1]*len(cover)` bueltatu behar du. Oraindik posiblea bada, oraindik prozesatu ez diren bi u eta v nodo topatu behar ditu. Bai u eta bai v existitzen ez badira cover osaturik dagoelako da eta cover bueltatu behar da. Nodo bakarra geratzen bada prozesatu gabe, cover-aren parte izango den ala ez erabaki behar da. Behin eginda, cover osoa bueltatu behar da. Bestela, u eta v aukeratu eta funtzioan jada idatzita dagoen kode zatiarekin jarraitu behar du. Zati hau ezin da aldatu.

Aurretik `lab06_` vertex_cover_2.py fitxategian inplementatu dituzun funtzioak erabili eta `lab06_vertex_cover_3.py`-n kopiatu. Adibidez, `partial_validity_check` funtzioa beharko duzu.

4. **Exekuzio denbora**

   Konparatu `vertex_cover_tree` funtzioaren exekuzio denborak zuhaitz bitarra eta zuhaitz hirutarra sortzen duten `recursive_vertex_cover`-en bertsioekin.

   Ze bertsiok tardatzen du gehiago? Zergatik?

5. **Inplementatu zuhaitz hirutar optimizatua**

   Aldagai bat false-era jartzen dugunean (nodo urdin bat) bere auzokide guztiak berdeak izan behar direla dakigu. Zure grafo hirutarraren kodea hartu ego-era hori isladatzeko. Konparatu exekuzio denbora aurreko algoritmoarekin eta ondorioak atera.

6. **Sakoneran ibilbidea zabalerako ibilbidearekin alderatuta**

   Laborategi honetan vertex-cover minimoa topatzeko bilaketa zuhaitzak erabili dituzu. Zuhaitz bakoitzan egindako ibilbidea sakoneran izan da. Ibilbidea zabaleran egingo bazenu, eraginkortasunean irabaziko zenuela uste duzu?