Розділ 1: Задача розмітки на ациклічних структурах
Домашнє завдання

Перш ніж почати

Вимоги

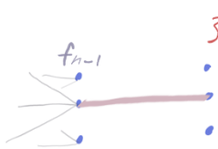
- якомога більше дій повинно бути описано словами: це дозволяє перевірити розуміння матеріалу та перевіряти роботи на унікальність;
- для кожної змінної, константи і функції повинно бути вказано множину (або область значень і визначення), якій вона належить: це дозволяє запобігати багатьох помилок та покращує зрозумілість роботи;
- письмові роботи виконуються індивідуально.

Рекомендації

- ретельно перевіряйте свою роботу перед тим, як відправляти її викладачеві;
- якщо завдання виконується на аркуші паперу і фотографується, намагайтеся задіяти достатньо освітлення і фотографувати якомога рівніше, щоб усі частини сторінки з розв'язком було добре видно; для спрощення цього процесу рекомендовано використовувати програми для смартфона, що мають можливість автоматичного пошуку документів на зображенні та обробки цих знімків на кшталт Adobe Scan, Google Drive, Dropbox, Fast Scan.

Ілюстрації

Рекомендації до виконання письмових завдань на прикладах і картинках



Звідки такий висновок?

$\Rightarrow d_{n-1} \in \arg\min_{d} f_{n-1}(d) = F_{n-1}(d)$

колізія імені d

d ← в якій множині шукати d?

Малюнок не є доведенням

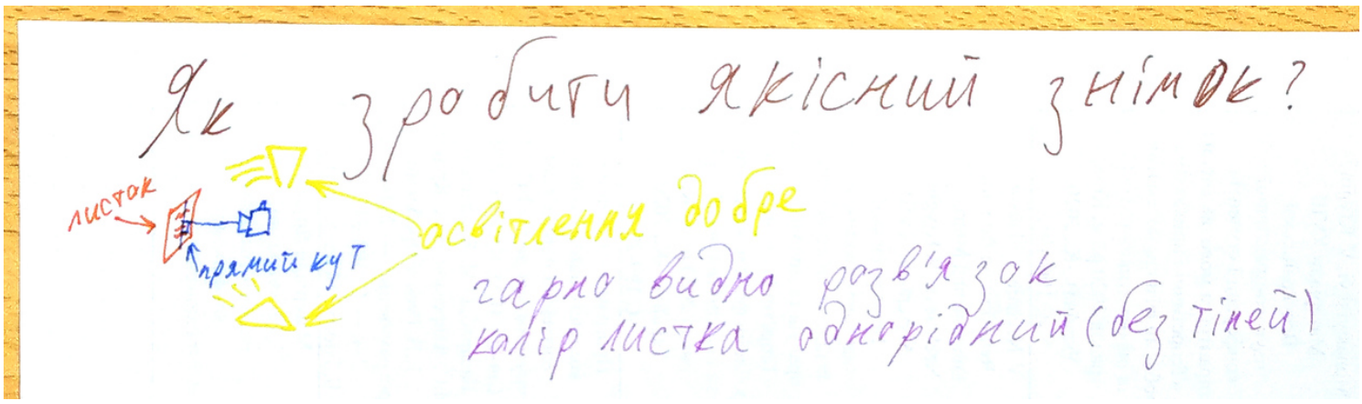
$\min_{d \in D} \left[f_{n-1}(d) + \lambda(d_n) \right] =$

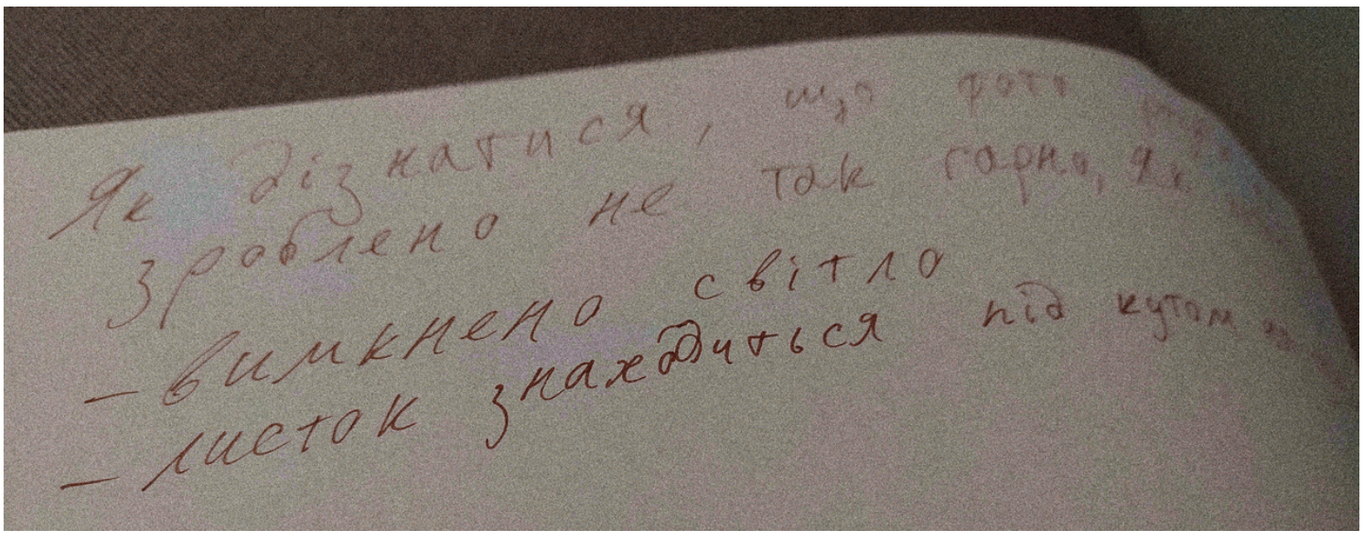
Оскільки $\lambda(d_n)$ не залежить від d, його можна винести як константу за межі мінімізації

детальні коментарі

$= \left[\min_{d \in D} f_{n-1}(d) \right] + \lambda(d_n), \quad \forall d_n \in D, \forall n = N..1.$

вказано множини визначено всі змінні





1 Основне (2 бали)

1.1 Задача розмітки на довільному напівкільці

Для довільного напівкільця $\langle R, \oplus, \otimes \rangle$, відомих скінченних множин T і K і функції $g : \{1, \ldots, n\} \times K^2 \to R$ вивести ефективний алгоритм пошуку значення

$$G = \bigoplus_{k \in K^T} \bigotimes_{i=1}^{n} g_i \left(k_{i-1}, k_i\right)$$

та розрахувати його складність. Навести приклади можливих задач із визначенням напівкільця та множин T і K.

1.2 Мінімізація часткового ризику

Показати, що для таких стратегій q^*, що

$$q^* \in \operatorname*{argmin}_{q: X \to D} \sum_{\substack{k \in K \\ x \in X}} w\left(q\left(x\right), k\right) \cdot p\left(x, k\right),$$

виконується

$$\forall x \in X : q^*\left(x\right) \in \operatorname*{argmin}_{d \in D} \sum_{k \in K} w\left(d, k\right) \cdot p\left(x, k\right).$$

Навіщо цей факт потрібен на практиці?

2 Додаткове (3 бали)

2.1 Задача розмітки на довільному напівкільці та ациклічній структурі

Для довільного напівкільця $\langle R, \oplus, \otimes \rangle$, відомого орієнтованого дерева $\langle T, \tau \rangle$, відомої скінченної множини K і функції $g : \tau \times K^2 \to R$ вивести ефективний алгоритм пошуку значення

$$G = \bigoplus_{k \in K^T} \bigotimes_{tt' \in \tau} g_{tt'}\left(k_t, k_{t'}\right)$$

та розрахувати його складність.

3 Комп'ютерне (4 бали)

Перш ніж почати

- спочатку розв'язується теорія, а вже потім на її основі пишеться код: це дозволяє не починати роботу з реалізації алгоритму, що є невірнім;
- бажано перевірити алгоритм на папірці: отримані дані також можна буде використовувати в автотестах;
- мова програмування довільна;
- дозволяється написання коду в бригадах розміром до двох людей; за commit messages буде визначатися внесок кожного учаснику, і на основі цього робитися висновок щодо того, чи було роботу виконано бригадою чи однією людиною;
- бажано використовувати технології, що спрощують збірку й запуск коду: CMake для C, C++ і Fortran, setuptools для Python, npm для NodeJS, Maven/Ant для Java/Scala і так далі.

Критерії оцінювання

- 2 бали (обов'язкова умова): коректно працююча програма з кодом у доступному викладачеві репозитарії (наприклад, GitHub, GitLab, BitBucket) та наявність теоретичного розв'язку поставленої задачі (теорія здається кожним учасником бригади окремо; вимоги ті ж самі, що й до основного завдання);
- 1 бал: наявність автотестів (наприклад, doctest у Python, Boost.Test у C++, JUnit у Java);
- 1 бал: наявність змістовних commit messages та CI (наприклад, GitHub Actions, GitLab CI, BitBucket Pipelines, TravisCI, Jenkins).

Якщо не знаєте, що обрати для тестування або CI, або як писати тести чи прив'язати до свого репозитарію CI, зверніться до викладача.

3.1 Розпізнавання тексту

Задача

Беремо англійську абетку (з пробілом) T та бінарні еталонні зображення букв $G : T \to \{0,1\}^{n \times m}$. Зображення тексту генерується шляхом конкатенації відповідних еталонів по горизонталі та незалежною інверсією кожного пікселя з ймовірністю p, яку вказує користувач. Ймовірності біграм, а також ймовірність для кожної букві знаходитися на першому місці в реченні дані.

Мета

Закріпити навички розпізнавання станів прихованих моделей Маркова.

Завдання

На вхід програмі подається

- шлях до папки, що містить еталони букв (еталони мають назви a.png, b.png, ..., z.png та space.png для пробілу);
- рівень шуму від 0 до 1;
- зашумлене зображення з однією строкою тексту.

Програма виводить розпізнаний текст.

Ймовірності надає викладач. Також програма повинна містити режим розпізнавання за рівномірних ймовірностей переходів (незалежне розпізнавання кожного символу).