ЗАВДАННЯ XXV МІЖНАРОДНОЇ ОЛІМПІАДИ З ІНФОРМАТИКИ ТА РЕКОМЕНДАЦІЇ ЩОДО ЇХ РОЗВ'ЯЗАННЯ

Продовження, початок в №7 за 2013 рік

Гуржій Андрій Миколайович,
віце-президент НАПН України, доктор технічних наук, професор, академік НАПН України.
Бондаренко Віталій Вікторович,
асистент факультету кібернетики Київського Національного університету ім. Тараса Шевченка.

РЕКОМЕНДАЦІЇ ЩОДО РОЗВ'ЯЗАННЯ ЗАВДАНЬ ПЕРШОГО ТУРУ

1. МАРЕННЯ

Звісно, що описана в умові структура озер і стежок задає дерево або кілька дерев (при $M<N-1$). У розв'язанні задачі будемо використовувати центр дерева — це вершина, від якої максимальна відстань до інших вершин є мінімальною серед всіх вершин.

Розглянемо випадок, коли $M=N-2$, тобто є два дерева і треба додати рівно одне ребро. При маленьких N можна просто перебрати можливі варіанти. Або, для більших N, можна помітити, що ребром треба з'єднати центри цих дерев.

У загальному випадку кількість дерев може бути більша за 2. Відсортуємо всі центри дерев за максимальною відстанню до вершин дерева. Нехай X — центр з максимальною відстанню. Приєднаємо решту центрів до нього та знайдемо максимальну відстань в отриманному дереві. Це можна зробити за два використання пошуку в глибину. Запустимо пошук з довільної вершини, знайдемо саму віддалену, другим пошуком знайдемо відстань до самої віддаленої до неї, це і буде відповідь. Складність такого розв'язку $O(N)$.

Щоб отримати вказану складність, треба вміти знаходити центр дерева за лінійний час. Просте рішення, яке за квадратичний час, перебирає всі вершини та знаходить відстані від них до всіх інших, не задовольняє обмеження по часу. Розв'яжемо таку задачу: для всіх вершин знайдемо довжину максимального шляху до цієї вершини. Ця задача розв'язується достатньо простим динамічним програмуванням по дереву. Маючи розв'язок цієї задачі, знайти центри можно, просто переглянувши всі вершини.

2. УРОКИ МИСТЕЦТВА

Ця задача має багато різних розв'язків. Розглянемо одне з найбільш простих.

Поглянемо на стилі картин, щоб зрозуміти, чим вони можуть відрізнятись. Може здаватись, що стиль картини можна визначити за кількісттю компонентів на заданому рисунку. Один компонент — це неперервна область близьких кольорів. Але цього не достатньо, такий розв'язок набере не більше половини балів.

Після того, як ми знашли кількість компонентів, ще треба порахувати кількість різних кольорів на кожній картині. Якщо підібрати відповідні константи, розв'язок отримає всі бали.

3. ВОМБАТИ

У цій задачі задано зважену решітку доріг (рядків багато, стовпчиків мало), по якій можна пересуватись вліво, вправо і вниз. Треба відповідати на багато запитів про знаходження найкоротшого шляху та на невелику кількість запитів про зміну ваги.

Перші три блоки тестів допускають простий розв'язок, для кожного запиту escape(), просто будемо шукати відповідь за даними про решітку доріг та кількість вомбатів.

Наступний блок тестів, де додатковим обмеженням є $C=2$, вимагає кращого розв'язку. Треба побудувати граф та за допомогою динамічного програмування або алгоритма Дейкстри шукати шлях найменшої довжини. Але цього виявляється недостатнім, зважаючи на співвідношення кількостей різних запитів, треба після кожної зміни кількості вомбатів перераховувати відповіді для всіх пар. Тоді обчислення відповіді на запит escape() буде виконуватись за константний час.

Повний розв'язок цієї задачі використовує розбиття задачі кореневою декомпозицією на блоки по 300 (це зменшить складність розв'язку у 300 разів). Тепер для кожного блоку будем окремо робити попередні підрахунки, а при внесенні змін нові обчислення потрібно робити тільки в одному блоку. На кожен із запитів про обчислення відстані нам потрібно буде пройти по всіх блоках та підрахувати відповідь.

ЗАВДАННЯ ДРУГОГО ТУРУ

1. ПЕЧЕРА

Під час пошуку місця проведення змагань — UQ Центру, ви заблукали і опинились у секретній групі печер під університетом. Вхід у групу печер перегороджує система безпеки, що включає в себе N дверей, розташованих одна за одною, і N перемикачів, кожен з яких з'єднано тільки з однією з дверей. Різні перемикачі з'єднані з різними дверима.

Двері пронумеровано від входу в групу печер від 0 до $(N-1)$ у порядку від ближніх до дальніх. Перемикачі також пронумеровані від 0 до $(N-1)$, однак, ви не знаєте, з якими дверими з'єднано кожен перемикач.

Усі перемикачі розташовано біля входу до групи печер. Кожен перемикач може бути в одній з двох позицій: «вверх» або «вниз». Одна з цих позицій є правильною. Якщо він знаходиться у правильній позиції, то з'єднані з перемикачем двері будуть відчинені, інакше ці двері буде зачинено. Правильна по-

зиція може відрізнятись для різних перемикачів, і ви не знаєте, яка з позицій для кожного перемикача є правильною (рис. 1).

Ви бажаєте зрозуміти, як побудовано систему безпеки. Для цього ви можете задати комбінацію позицій перемикачів, тобто встановити кожен з перемикачів у довільну позицію; після чого зайти до печери і дізнатись номер перших зачинених дверей. Двері, що знаходяться за зачиненими дверима, не видно.

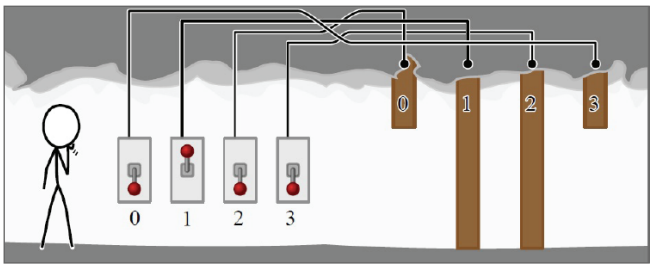

Рис. 1

У вас є час на те, щоб спробувати не більше 70 000 комбінацій позицій перемикачів. Ваша задача — визначити правильну позицію для кожного перемикача, а також для кожного перемикача знайти двері, з якими він з'єднаний.

Деталі реалізації. Ваш розв'язок має реалізовувати функцію exploreCave(). Вона може викликати функцію модуля перевірки tryCombination() не більше 70 000 разів і повинна завершуватись викликом функції answer(). Ці функції описано нижче.

Функцію tryCombination(*S*[]) реалізує модуль перевірки. Вона дозволяє вам задати комбінацію позицій перемикачів для того, щоб зайти у групу печер і дізнатися номер перших зачинених дверей. Якщо всі двері відчинено, то функція повертає значення −1. Ця функція працює за час $O(N)$, тобто час роботи у гіршому випадку є пропорційним N.

Цю функцію можна викликати не більше 70 000 разів. Параметри:

- S — масив довжини N, що задає комбінацію позицій перемикачів. Елемент масиву $S[i]$ відповідає перемикачу з номером i. Значення 0 означає, що перемикач знаходиться у позиції «вверх». Значення 1 означає, що перемикач знаходиться в позиції «вниз»;
- значення, що повертається — номер перших зачинених дверей або −1, якщо всі двері відчинено.

Функцію answer(*S*[], *D*[]) також реалізує модуль перевірки. Викличте цю функцію, коли визначено комбінацію позицій перемикачів, при якій всі двері відчинені, а також для кожного перемикача визначено двері, з якими він з'єднаний.

Параметри:

- S — масив довжини N, що містить правильну позицію кожного з перемикачів. Формат цього масиву такий самий, як і у параметра описаної вище функції tryCombination();
- D — масив довжини N, елементи якого для кожного перемикача задають, з якими дверима він з'єднаний. Елемент $D[i]$ має містити номер дверей, з якими з'єднано перемикач з номером i;
- значення, що повертається: ця функція не повертає керування вашій програмі, тобто приводить до завершення вашої програми.

Ваш розв'язок має реалізовувати функцію exploreCave(N). Ця функція має використовувати функцію модуля перевірки tryCombination(), щоб визначити правильну позицію для кожного перемикача, а також для кожного перемикача визначити двері, з якими він з'єднаний. Коли ваша функція отримає цю інформацію, вона має викликати функцію answer().

Параметри:

- N — кількість перемикачів і дверей у групі печер.

Приклад. Припустимо, що двері і перемикачі розташовані як показано на рис. 1. Можливий порядок виклику функцій подано у таблиці 1.

Обмеження

- $1 \le N \le 5000$.

Підзадачі (табл. 2)

Рекомендації щодо розв'язання

З обмежень в умові задачі видно що на кожні двері можно в середньому використовувати 14 запитів, і 2^{14} трохи перевищує кількість дверей, отже обмеження дозволяють скористатить бінарним пошуком.

Почнемо з перших дверей та будемо з'ясовувати по черзі який перемикач відчиняє чергові двері та його положення при цьому. Як тільки ми знайшли цей перемикач, зафіксуємо його у правильній позиції і більше не будемо його чипати, перейдемо до наступних дверей.

Щоб знайти перемикач для поточних дверей, почнемо з усіх незафіксованих перемикачей у однаковій позиції. Цей запит дозволить нам одразу дізнатись правильну позицію перемикача, пов'язаного з поточними дверима. Далі будемо змінювати позицію половини перемикачів та перевіряти, чи змінився стан дверей, тим самим виключаючи з розгляду половину перемикачей. Будемо так діяти до тих пір, поки не залишиться один перемикач. Складність алгоритма $O(N \log N)$, гарантується не більше $N([\log N]+1)$ запитів.

2. РОБОТИ

Маленький брат Марити розкидав іграшки по всій кімнаті! Дуже доречно, Марита розробила спеціальних роботів для прибирання кімнат. Вона просить вас допомогти визначити, які іграшки має прибирати кожен з роботів.

Таблиця 1

Виклик функції	Повертається	Пояснення
tryCombination([l, 0, 1, 1])	1	Ситуація відповідає показаній на рис. 1. Перемикачі 0, 2 і 3 знаходяться в позиції "вниз а перемикач 1 — в позиції "вверх". Функція повертає значення 1, що означає, що двері 1 — перші двері ліворуч, що є зачиненими.
tryCombination([0, 1, 1, 0])	3	Двері 0, 1 і 2 відчинені, а двері 3 — зачинені.
tryCombination([l, 1, 1, 0])	-1	Переключення перемикача 0 у позицію "вниз "призводить до того, що всі двері відчинились, що позначається повернутим значенням -1.
answerQl, 1, 1, 0],[3, 1, 0, 2])	(нема)	Ваша програма визначила, що комбінація [1, 1, 1, 0] задає правильні позиції кожного з перемикачів, і перемикачі 0, 1, 2 і 3 з'єднані з дверима 3, 1, 0 і 2, відповідно.

Таблиця 2

Підзадача	Бали	Додаткові обмеження на вхідні дані
1	12	Для кожного i, перемикач з номером i з'єднаний з дверима з номером i. Ваша задача — визначити правильні позиції для кожного перемикача
2	13	Правильна позиція всіх перемикачів завжди [0, 0, 0, . . . , 0]. Ваша задача — визначити, який перемикач з'єднаний з якими дверима
3	21	$N \leq 100$
4	30	$N \leq 2000$
5	24	(немає)

- Всього розкидано T іграшок, кожна іграшка має цілочисельну вагу $W[i]$ і цілочисельний розмір $S[i]$. Роботи бувають двох типів: слабкі і маленькі.
- Усього є A слабких роботів. Кожен слабкий робот має деяке обмеження за вагою $X[i]$, тобто він може пересувати іграшки ваги строго менше ніж $X[i]$. Розмір іграшки при цьому не має значення.
- Усього є B маленьких роботів. Кожен маленький робот має деяке обмеження по розміру $Y[i]$, тобто він може пересувати іграшки розміру строго менше ніж $Y[i]$. Вага іграшки при цьому не має значення.

Процес прибирання кожної іграшки для довільного робота займає одну хвилину. Кожен робот може прибирати тільки одну іграшку за хвилину. Різні роботи можуть одночасно прибирати різні іграшки.

Ваша задача полягає у тому, щоб визначити, чи можливо прибрати всі іграшки, і якщо так, обчислити мінімальний час, за який це можна зробити.

Приклади. Припустимо, що у Марити є A=3 слабких роботів з обмеженнями по вазі X=[6,2,9], B=2 маленьких роботів з обмеженнями по розміру Y=[4, 7] і T=10 іграшок з параметрами, що описано у таблиці 3.

Таблиця 3

Номер іграшки	0	1	2	3	4	5	6	7	8	9
Вага	4	8	2	7	1	5	3	8	7	10
Розмір	6	5	3	9	8	1	3	7	6	5

Мінімальний час, за який можна прибрати всі іграшки, — три хвилини. Один з оптимальних способів розподілу роботи між роботами показано в таблиці 4.

Таблиця 4

	Слабкий робот 0	Слабкий робот 1	Слабкий робот 2	Маленький робот 0	Маленький робот 1
Перша хвилина	Іграшка 0	Іграшка 4	Іграшка 1	Іграшка 6	Іграшка 2
Друга хвилина	Іграшка 5		Іграшка 3		Іграшка 8
Третя хвилина			Іграшка 7		Іграшка 9

Розглянемо інший приклад. Припустимо, що у Марити є A=2 слабких роботів з обмеженнями по вазі X=[2,5], B=1 маленький робот з обмеженням по розміру Y=[2], а також T=3 іграшки з параметрами, що описано у таблиці 5. Жоден з роботів не може прибрати іграшку вагою 5 і розміром 3, тому не існує способу прибрати всі іграшки.

Деталі реалізації. Ваш розв'язок має реалізовувати описану нижче функцію putaway(A, B, T, X[], Y[], W[], S[]).

Ця функція має обчислювати мінімально можливий час у хвилинах, який потрібно роботам, щоб прибрати усі іграшки, або повертати –1, якщо не існує жодного способу прибрати всі іграшки.

Таблиця 5

Номер іграшки	0	1	2
Вага	3	5	2
Розмір	1	3	2

Параметри:

- A — кількість слабких роботів.
- B — кількість маленьких роботів.
- T — кількість іграшок.
- X — масив довжини A, який містить цілі числа, кожне з яких описує обмеження по вазі для відповідного слабкого робота.
- Y — масив довжини B, який містить цілі числа, кожне з яких описує обмеження по розміру для відповідного маленького робота.
- W — масив довжини T, який містить цілі числа, кожне з яких описує вагу відповідної іграшки.
- S — масив довжини T, який містить цілі числа, кожне з яких описує розмір відповідної іграшки.
- Значення, що повертається: мінімально можливий час, який потрібно роботам, щоб прибрати усі іграшки, або –1, прибрати іграшки неможливо.

Приклад. Наступний набір даних описує перший приклад, наведений вище в умові задачі (табл. 6).

Таблиця 6

Параметр	Значення
A	3
B	2
T	10
X	[6, 2, 9]
Y	[4, 7]
W	[4, 8, 2, 7,1, 5, 3, 8, 7, 10]
S	[6, 5, 3, 9, 8, 1, 3, 7, 6, 5]
Результат	3

Наступний набір даних описує другий приклад, наведений вище в умові задачі (табл. 7).

Таблиця 7

Параметр	Значення
A	2
B	1
T	3
X	[2, 5]
Y	[2]
W	[3, 5, 2]
S	[1, 3, 2]
Результат	–1

Обмеження

- $1 \leq T \leq 1\,000\,000$;
- $0 \leq A, B \leq 50\,000$ та $1 \leq A+B$;
- $1 \leq X[i], Y[i], W[i], S[i] \leq 2\,000\,000\,000$.

Підзадачі (табл. 8)

Таблиця 8

Підзадача	Бали	Додаткові обмеження на вхідні дані
1	14	$T = 2$ *і* $A + B = 2$ (рівно два робота і рівно дві іграшки)
2	14	$B = 0$ (всі роботи — слабкі)
3	25	$T \le 50$, $A + B \le 50$
4	37	$T \le 10000$, $A + B \le 1000$
5	10	(немає)

Рекомендації щодо розв'язання

Іграшки неможна прибрати тоді і тільки тоді, коли існує іграшка з більшою вагою, ніж може підняти найсильніший із слабких роботів і одночасно з більшим розміром, ніж може взяти маленький робот з найбільшим обмеженням по розміру. Інакше іграшки можна прибрати принаймні за час, що дорівнює кількості іграшок.

Досить часто в задачах, де потрібно знайти мінімальний час для досягнення мети, можна скористатися бінарним пошуком по відповіді, отже нам потрібно знайти метод перевірки чи можуть роботи прибрати кімнату за S хвилин.

Спочатку визначимо дію для слабких роботів, почавши з найслабкішого. Є деякий набір іграшок, що він може прибрати, і важливою залишається тільки інформація про те, який розмір мають ці іграшки. Очевидно, найслабкіший робот має починати прибирання з найбільших іграшок, щоб полегшити роботу маленьких роботів у подальшому. Також, немає сенсі призначати прибирання меньше ніж S іграшок, крім випадку коли іграшки відповідное ваги скінчились.

Тепер розглянемо другого по силі робота. Є додатковий набір іграшок, які він може прибирати, а також деякі легші іграшки на які у першого робота не вистачило часу. Оскільки найслабіший робот вже має призначення, вага перестає бути суттєвою і треба знову обирати іграшки у порядку зменшення їх розміру. Той самий процес повторюється для решти слабких роботів, у порядку збільшення її обмеження по вазі.

Тепер розглянемо маленьких роботів, від найбільшого до найменшого. Кожен з них має взяти щонайбільше S іграшок, починаючи з найбільшое, що залишились. Якщо черговий робот не може прибрати нійбільшу іграшку, то S виявилось занадто малим.

Реалізувати цей алгоритм можна використовуючи чергу з приорітетами, реалізовану як бінарна куча, що буде містити іграшки, достатньо легкі щоб бути прибраними поточним роботом, та впорядковану з найбільшою іграшкою на початку. Іграшки спочатку відсортовані за вагою. Кожного разу, коли ми переходимо до нового робота, ми додаємо нові елементи зі списку іграшок до черги, після чого робот видаляє їх починаючи з голови черги.

Складність алгоритма буде $O(N\log N+(A+B)+\log N(N\log(A+B)))=O(N\log N\log(A+B))$.

3. ГРА

Базза і Шазза грають у гру. Дошка для цієї гри є прямокутною таблицею, що містить R рядків з номерами від 0 до $R-1$ і C стовпчиків з номерами від 0 до $C-1$. Позначимо (P, Q) клітину таблиці на перетині рядка P і стовпчика Q. У кожній клітині записано невід'ємне ціле число. На початку гри у всіх клітинах записано нулі. Гра проходить так. Кожним ходом Базза може:

- присвоїти нове число клітині (P, Q);
- або задати Шаззі запитання, чому дорівнює найбільший спільний дільник (НСД) всіх цілих чисел з клітин всередині прямокутника, протилежні кути якого — клітини (P, Q) та (U, V), включно.

Базза робить не більше (N_U+N_Q) ходів (N_U — кількість присвоювань чисел клітинам, а N_Q — кількість запитань), а потім зморюється і йде грати в крикет.

Ваша задача — визначити правильні відповіді на запитання.

Приклади. Припустимо, що R=2 і C=3, і Базза починає гру з таких ходів:

- присвоює клітині (0, 0) число 20;
- присвоює клітині (0, 2) число 15;
- присвоює клітині (1,1) число 12.

Отриману таблицю показано на рисунку 2. Базза далі може задати запитання, чому дорівнює НСД всередині таких прямокутників:

- з протилежними кутами (0, 0) і (0, 2) (у цьому прямокутнику три числа — 20, 0 і 15, і їх НСД дорівнює 5);
- з протилежними кутами (0, 0) і (1, 1) (у цьому прямокутнику чотири числа — 20, 0, 0 і 12, і їх НСД дорівнює 4).

Припустимо, що тепер Базза робить наступні ходи:

- присвоює клітині (0, 1) число 6;
- присвоює клітині (1, 1) число 14.

Нову таблицю показано на рисунку 3. Базза далі може знову ставити запитання, чому дорівнює НСД всередині наступних прямокутників:

- з протилежними кутами (0, 0) і (0, 2) (тепер у цьому прямокутнику три числа — 20, 6 і 15, і їх НСД дорівнює 1);
- з протилежними кутами (0, 0) і (1, 1) (тепер у цьому прямокутнику чотири числа — 20, 6, 0 і 14, і їх НСД дорівнює 2).

У цьому прикладі Базза зробив всього N_U=5 присвоювання і задав N_Q=4 запитання.

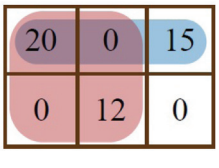

Рис. 2

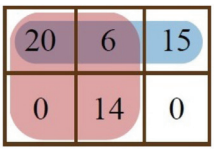

Рис. 3

Деталі реализації. Ваш розв'язок має реалізовувати описані нижче функції init(R, C), update(P, Q, K) і calculate(P, Q, U, V).

Щоб допомогти вам, кожен з шаблонів розв'язків на вашому комп'ютері містить функцію gcd2(X, Y), яка обчислює найбільший спільний дільник двох заданих цілих невід'ємних чисел X і Y.

Якщо X=Y=0, то функція gcd2(X, Y) повертає 0. Ця функція виконується досить швидко для того, щоб міг набрати повний бал, зокрема, час роботи цієї функції у гіршому випадку пропорційний $\log(X+Y)$.

init(R, C):

Ця функція задає вам розмір таблиці і дозволяє ініціалізувати довільні глобальні змінні і структури даних. Її буде викликано тільки один раз перед іншими викликами функцій update() чи calculate().

Параметри:

- R — кількість рядків;
- C — кількість стовпчиків. update(P, Q, K).

Ця функція буде викликатись, коли Базза присвоює число будь-якій клітині. Параметри:

- P — рядок, у якому знаходиться клітина ($0 \le P \le R-1$);
- Q — стовпчик, у якому знаходиться клітина ($0 \le Q \le C-1$);
- K — нове ціле число для цієї клітини ($0 \le K \le 10^{18}$). Воно може не відрізнятися від попереднього числа у цій клітині.

calculate(P, Q, U, V):

Ця функція має обчислювати найбільший спільний дільник всіх чисел у прямокутнику з протилежними кутами (P, Q) і (U, V), включно, тобто клітини (P, Q) і (U, V) включаються у прямокутник.

Якщо всі числа у цьому прямокутнику — нулі, ця функція має повертати нуль.

Параметри:

- P — рядок, у якому знаходиться верхня ліва клітина прямокутника ($0 \le P \le R-1$);
- Q — стовпчик, у якому знаходиться верхня ліва клітина прямокутника ($0 \le Q \le C-1$);
- U — рядок, у якому знаходиться нижня права клітина прямокутника ($0 \le U \le R-1$);
- V — стовпчик, у якому знаходиться нижня права клітина прямокутника ($0 \le V \le C-1$);
- значення, що повертається: значення НСД всіх чисел у цьому прямокутнику або 0, якщо всі числа у цьому прямокутнику рівні нулю.

Приклад. Наступна послідовність викликів функції (табл. 9) відповідає прикладу, що наведено вище в умові задачі.

Обмеження

- $1 \le R, C \le 10^9$;
- $0 \le K \le 10^{18}$, де K довільне з чисел, які Базза присвоює в клітини таблиці.

Підзадачі (табл. 10)

Таблиця 9

Виклик функції	Значення, що повертається
init(2, 3)	
update(0, 0, 20)	
update(0, 2, 15)	
update(1, 1, 12)	
calculate(0, 0, 0, 2)	5
calculate(0, 0, 1, 1)	4
update(0, 1, 6)	
update(1, 1, 14)	
calculate(0, 0, 0, 2)	1
calculate(0, 0, 1, 1)	2

Таблиця 10

Бали	R	C	N_u	Q_u
10	≤ 100	≤ 100	≤ 100	≤ 100
27	≤ 10	≤ 100 000	≤ 10 000	≤ 250 000
26	≤ 2000	≤ 2000	≤ 10 000	≤ 250 000
17	≤ 10^9	≤ 10^9	≤ 10 000	≤ 250 000
20	≤ 10^9	≤ 10^9	≤ 22 000	≤ 250 000

Рекомендації щодо розв'язання

Задача очевидно зводиться до реалізації двовимірної розрідженої структури даних, що дозволяє рахувати значення функції НСД на прямокутнику за $O(\log(S))$, де S — число елементів в структурі, $S \le N_U$. Нулі можна ігнорувати, оскільки вони не впливають на значення НСД. Таким чином, складність алгоритму буде $O((N_Q+N_U)\log(N_U)\log(Z))$, де Z — максимальне значення чисел у структурі.

Можливі реалізації — двовимірне неявне дерево відрізків або дерево відрізків декартових дерев.

★ ★ ★

ВИКОРИСТАННЯ СЕРЕДОВИЩА SCRATCH ДЛЯ ПІДГОТОВКИ УЧНІВ ДО ОЛІМПІАДИ З ПРОГРАМУВАННЯ

Петриченко Тетяна Миколаївна,

учитель інформатики вищої кваліфікаційної категорії, старший вчитель Уманської спеціалізованої школи I–III ступенів №12 з поглибленим вивченням англійської мови.

Сучасне суспільство ставить перед освітою складне завдання: підготувати спеціаліста, який володіє не тільки певним багажем знань, але й здатного до постійного самовдосконалення, самоосвіти й адаптації до нових вимог. Тому суспільство потребує прориву в підготовці інтелектуальної молоді. Сучасна школа має не тільки передати знання, а й навчити учня їх здобувати й уміти використовувати.

Багато фахівців працюють у галузі інформаційних технологій і намагаються привернути увагу школярів до наукової діяльності, причому з раннього віку. Так з 2008 року проводиться Міжнародний конкурс з інформатики і комп'ютерної вправності Бобер. Учні 2–11-их класів мають змогу взяти участь і проявити власні здібності в розв'язуванні цікавих інтерактивних задач.

Першим кроком у підготовці учнів до олімпіади з програмування — залучення їх до участі у конкурсах ще з раннього віку. Щоб підготувати дитину до олімпіади, треба серйозно займатися, починаючи з початкової й основної школи.

Тому другий крок — зацікавити.

Розв'язання цієї проблеми може базуватися на використанні у навчанні інформатики мови програмування Scratch.Середовище Scratch є оптимальним вибором для вказаної вікової групи і сприяє формуванню пізнавальної, інформаційної і комунікативної компетенції учнів. У Scratch поєднані ідеї програмування, властиві середовищу Лого і Лего-Лого. Але, тепер вони втілені на більш високому рівні. Користувачі можуть збирати свої програми-процедури з блоків так само, як вони збирали конструкції з цеглинок Лего (рис.1).Це середовище забезпечує баланс між інтенсивним інтелектуальним і психологі-