Кіріспе

Таңдалған тақырыптың өзектілігі деректер құрылымдары мен алгоритмдер программалық жасақтама жасалатын материалдар болып табылады.Көбінесе кіріс деректерінің үлкен көлемімен жұмыс істейтін қолданбаларды әзірлеу кезінде программаны орындау кезінде оларды сақтау туралы сұрақ туындайды. Мәліметтер массиві деректерді сақтау мәселесін шешетіні сөзсіз, бірақ оның кемшіліктері де жоқ емес екені анық. Ең бастысы, әрине, бекітілген өлшем. Бұл сипатты тіпті динамикалық түрде жасалған массивтер үшін өзгерту мүмкін емес, сондықтан олар үшін «резерві бар» жадты бөлу керек. Дегенмен, «резерв» де шектеулі және оның жеткілікті болатынына ешкім кепілдік бере алмайды, әйтпесе бұл «резерв» программаға бөлінген жадтың көп бөлігі босқа кетуі үшін жеткілікті болуы мүмкін.Бұл мәселе динамикалық деректер құрылымдарының көмегімен шешіледі.

Динамикалық айнымалылар динамикалық жадта программаның орындалу барысында құрылады. Оларға көрсеткіштер арқылы қол жеткізіледі. Осы айнымалы мәндерді пайдалана отырып, программаны іске қоспас бұрын белгісіз деректерді өңдеуге болады. Мұндай деректерге арналған жады бір-бірімен әрекеттесетін блоктарға бөлінген. Мәліметтерді сақтаудың бұл әдісі динамикалық құрылымдар деп аталады.

Ағаштар белгілі сызықты емес құрылымдар. Олар деректерді сызықтық түрде сақтамайды, бірақ оны иерархиялық түрде ұйымдастырады. Ағаштар әртүрлі ақпарат түрлерін ұсынуға өте ыңғайлы құрал.

Ағаштардың қарапайым графиктерден айырмашылығы ағашты кесіп өткенде циклге өту мүмкін емес. Бұл графиктерді әртүрлі алгоритмдер үшін деректерді ұйымдастырудың өте ыңғайлы формасына айналдырады. Осылайша, ағаш түсінігі информатикада және динамикалық деректер құрылымдарын зерттеуде программалауда белсенді қолданылады.

Зерттеу объектісі деректер құрылымдары болып табылады.

Зерттеу пәні - бинарлы ағаштар.

Зерттеудің мақсаты динамикалық деректер құрылымдарын, атап айтқанда екілік ағаштарды зерттеу болып табылады.

Зерттеудің мақсаттары оның мақсатына қарай құрылады және келесідей:

- мәліметтердің динамикалық құрылымдарын қарастыру;

- екілік ағаштағы амалдарды қарастыру.

1 Пәндік аймақты зерттеу

1.1 Екілік ағаштар

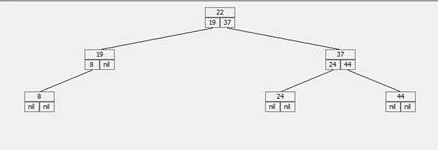
Деректер ағаштары программалаудағы алгоритмдер мен деректер құрылымдарының негізгі элементтерінің бірі болып табылады. Ағаштардың көптеген түрлерінің ішінде екілік іздеу ағашы (Binary Search Tree, BST) ақпаратты тиімді сақтау және өңдеу үшін кеңінен қолданылатын негізгі деректер құрылымдарының бірі болып табылады. BST элементтерді кірістіру, жою және іздеу сияқты операцияларды қолдайды, оларды орташа логарифмдік уақытта орындайды, бұл оны көптеген қосымшаларға ыңғайлы етеді.

Деректер мен ақпарат шешуші рөл атқаратын қазіргі әлемде алгоритмдерді түсініп қана қоймай, оларды елестете білу де маңызды. Визуализация алгоритмдер мен деректер құрылымдарының қалай жұмыс істейтінін тереңірек түсінуге көмектеседі, оқуды жеңілдетеді және күрделі есептерді шешуге көмектеседі.

Екілік іздеу ағаштары информатиканың әртүрлі салаларында кең ауқымды қолданбаларға ие. Олар индекстерді тиімді іздеу және басқару үшін дерекқорларда, деректерді сақтауды оңтайландыру үшін файлдық жүйелерде және қысу және шифрлау алгоритмдерінде қолданылады. Реттелген деректерді сақтау және элементтерді жылдам енгізуді, жоюды және іздеуді қамтамасыз ету қабілетінің арқасында екілік іздеу ағаштары үлкен көлемдегі ақпаратты өңдеу тапсырмаларында таптырмас болып табылады.

Екілік ағаш - ағаштың түбірі деп аталатын бос немесе құрамында бір элементі бар элементтердің ақырлы жиыны, ал жиынның қалған элементтері әрқайсысы екілік ағаш болып табылатын екі ажыратылған ішкі жиынға бөлінеді.[5] Бұл ішкі жиындар бастапқы ағаштың сол және оң ішкі ағаштары деп аталады. Екілік ағаштың әрбір элементі ағаш түйіні деп аталады.

1.1-суретте екілік ағашты бейнелеу тәсілі көрсетілген.



Бұл ағаш алты түйіннен, 22 тамырлы ағаштан тұрады. Оның сол жақ ішкі ағашында 19 түбір, ал оң ішкі ағашта 37 түбір бар. Бұл 22-ден шығатын екі тармақпен ұсынылған: сол жақтан 19-ға дейін және оң жақтан 37-ге дейін. Бұтақтың болмауы бос ішкі ағашты көрсетеді. Мысалы, 19 түбірі бар екілік ағаштың сол жақ ішкі ағашы бос. 37-де тамыры бар екілік ағашта бос сол және оң ішкі ағаштар бар.

Егер А екілік ағаштың түбірі, ал В оның сол немесе оң ішкі ағашының түбірі болса, онда А - В әкесі, ал В - А-ның сол немесе оң ұлы. Ұлдары жоқ түйін (мұндай 8, 24 және 44) түйіндері ретінде жапырақ деп аталады.

Мысалы, 1.3-суреттегі ағашта: 22 - 19-дың атасы және 8 саны - 19-ның баласы, бірақ 8 - 22-нің атасы да, баласы да емес.

Екі түйін бір әкенің ұлдары болса, ағайынды болады. Екілік ағаштың әрбір жапырақты емес түйінінде бос емес оң және сол ішкі ағаштар болса, онда ағаш қатаң екілік ағаш деп аталады. n жапырағы бар қатаң екілік ағашта әрқашан 2n-1 түйін болады. Бұл жағдайда ол екілік ағаш емес.

Реттелген екілік ағаштар - әрбір Х шыңы үшін ереже орындалатын ағаштар: сол жақ ішкі ағашта кілттер Х-дан кіші, оң жақ ішкі ағашта кілттер Х-дан үлкен немесе оған тең. Екілік ағаштың құрылымы мынадан құрастырылған түйіндер. Ағаш түйіні деректер өрісін және екі көрсеткіш өрісін қамтиды.[7]

Элементті қосқанда, қосылған элемент ағашта бір орынды алатындай реттілік қасиетін сақтау керек (әрине, егер мұндай элемент ағашта жоқ болса). Ағаштағы элементтің болуы дәл осылай тексеріледі - түйіннен түйінге өту қажетті мәнге максималды жақындау принципі бойынша жүреді. Қажетті элемент болуы керек жерде ішкі ағаш болмаса, онда мұндай элемент жиынтықта жоқ деп айта аламыз.

1.2 Екілік ағаш түйіні

Delphi программасының екілік ағаш түйінін анықтау кезінде бізге оның еншілес түйіндеріне екі сілтеме (яғни, көрсеткіштер) және түйінде сақталатын нақты деректер (деректер бөлігі) қажет. Программа жұмыс істеп тұрған кезде ағаш құрылымын өзгертуге болады: түйіндерді қосу немесе жою, ақпараттық бөлікті өзгерту. Басқаша айтқанда, ағаш динамикалық құрылым. Ағаш түйінді ақпараттық бөлігі (мысалы, бүтін сан) және осы түйіннің сол және оң ішкі ағаштарына нұсқайтын екі сілтемесі бар бекітілген құрылымы бар айнымалы ретінде сипаттауға болады. Бұл үшін - record түрі қолайлы. Бос ағашқа сілтеме нөл болуы керек.

Ағаш түйінінің сипаттамасы келесідей болуы мүмкін[8]:

Type PTree = ^TTree; // ағаш түйініне көрсеткіш

TTree = record

Inf:integer; // ақпарат бөлігі (түрі кез келген болуы мүмкін және тапсырмаға байланысты)

Left,Right::PTree; // сол және оң ішкі ағаштарға нұсқау

End

2. Алгоритм және математикалық модель

2.1 Екілік ағаштағы амалдар

Бинарлы ағаштар (және жалпы ағаштар) үшін алгоритмдердің маңызды категориясы енгізілген - ағаштарды өту алгоритмдері. Бұл алгоритм ағаштың әрбір түйініне тек бір рет қол жеткізуге мүмкіндік беретін әдіс.

Әрбір түйін үшін ақпараттық бөлікті өңдеудің кейбір түрлері орындалады (тексеру, жинақтау және т.б.), бірақ айналып өту процедурасы түйінде орындалатын нақты әрекеттерге тәуелді емес және түйінді өңдеудің барлық алгоритмдеріне ортақ. Процесс түбірінде орындалатын әрекеттерді шақырайық. Ағаштарды айналдырудың әртүрлі әдістері белгілі бір ретпен жеке түйіндерге баруды қамтиды.

Тереңдік-бірінші өтуді қолданатын үш рет бар[9]:

1. Жоғарыдан төменге қарай (Алдын ала тапсырыс):

- түбірді өңдеу;

- сол бағыныңқы ағаштың өтуі;

- оң жақ бағыныңқы ағаштың өтуі;

2. Солдан оңға қарай (InOrder):

- сол бағыныңқы ағаштың өтуі;

- түбірді өңдеу;

- оң жақ бағыныңқы ағаштың өтуі;

3. Төменнен жоғары (PosOrder):

- сол бағыныңқы ағаштың өтуі;

- оң жақ бағыныңқы ағаштың өтуі;

- түбірді өңдеу.

Бұл үш әдісті рекурсивті процедуралар түрінде оңай ұсынуға болады, олардың тізімі төменде келтірілген. Барлық процедураларда түбірді өңдеу үшін экранда ақпараттық өрістің мәнін көрсету қолданылады. Екілік ағаштың түбірінің адресі Tree:P Tree параметрі ретінде процедураларға беріледі.

Екілік ағаш өту процедураларының тізімі.[10]

Procedure PrintTree(Tree: PTree); // PreOrder

begin

if Tree<>nilthen begin //

Writeln(‘Value = ‘,Tree^.inf); //

PrintTree(Tree^.Left); //

PrintTree(Tree^.Right); //

end;

end;

Procedure PrintTree(Tree: PTree); // InOrder

begin

if Tree<>nil then begin //

PrintTree(Tree^.Left); //

Writeln(‘Value = ‘,Tree^.inf); //

PrintTree(Tree^.Right); //

end;

end;

Procedure PrintTree(Tree: PTree); // PosOrder

begin

if Tree<>nil then begin //

PrintTree(Tree^.Left); //

PrintTree(Tree^.Right); //

Writeln(‘Value = ‘,Tree^.inf); //

end;

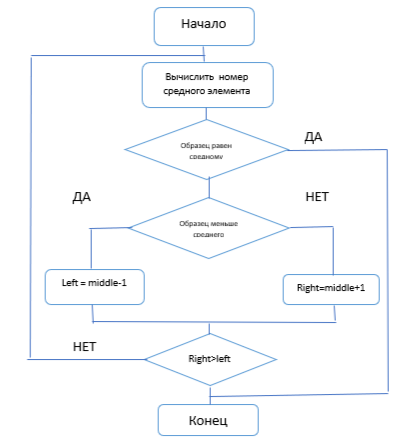
end;

Ағаш сілтемесі мән параметрі ретінде жіберіледі, яғни ол нақты параметрдің көшірмесін алатын жергілікті процедура айнымалысы болып табылады. Оны өзгерту нақты параметрді өзгертпейді (себебі процедура оның жергілікті көшірмесінде жұмыс істейді). Бұл ағаш түбірінің мекенжайы нақты параметр ретінде берілсе, оны сақтайтын айнымалы өзгеріссіз қалады дегенді білдіреді.

2.2 Екілік іздеу ағашының блок схемасы

Екілік іздеу ағашының блок-схемасы – екілік іздеу алгоритмінің графикалық түрде бейнеленуі, ағаш құрылымында элементті іздеу процессін көрсетеді.

Берілген (2.1-сурет) блок-схема екілік іздеу (binary search) алгоритмының жұмыс істеу принципін көрсетеді. Бұл алгоритм сұрыпталған тізімнен белгілі бір элементті іздеуге арналған және әр қадамда тізімді екіге бөліп, іздеуді жылдамдатады. Алгоритм “Бастау” блогынан басталады, мұнда іздеу процесі басталады. Алдымен тізімдегі ең сол жақ (Left) және ең оң жақ (Right) көрсеткіштердің орташа мәні есептеліп, сол арқылы ортаңғы элементтің индексі анықталады. Бұл ортаңғы элемент ізделіп жатқан мәнмен салыстырылады. Егер ізделіп жатқан элемент ортаңғы элементке тең болса, алгоритм аяқталып, ізделген элемент табылғаны анықталады. Егер ізделіп жатқан элемент ортаңғы элементтен кіші болса, бұл элемент тек тізімнің сол жақ бөлігінде орналасуы мүмкін. Сондықтан тізімнің оң жақ бөлігі алынып тасталады, ал іздеу аймағы сол жақ бөлігіне тарылып, Right = middle - 1 формуласы қолданылады. Егер ізделіп жатқан элемент ортаңғы элементтен үлкен болса, онда бұл элемент тек тізімнің оң жақ бөлігінде орналасуы мүмкін. Осыған байланысты тізімнің сол жақ бөлігі алынып тасталады, ал іздеу аймағы оң жақ бөлігіне тарылып, Left = middle + 1 формуласы қолданылады. Әр қадамнан кейін Left > Right шарты тексеріледі. Егер сол жақ шекара (Left) оң жақ шекарадан (Right) асып кетсе, бұл іздеудің аяқталғанын және элементтің табылмағанын білдіреді. Мұндай жағдайда алгоритм аяқталады. Алгоритм тізімді әр қадамда екіге бөліп, іздеу аймағын біртіндеп тарылтады. Егер тізімдегі элементтер саны n болса, алгоритм элементті логарифмдік жылдамдықпен, яғни log2(n) қадамда таба алады. Бұл әдістің басты ерекшелігі – оның тиімділігінде, бірақ ол тек сұрыпталған тізімдерде ғана жұмыс істейді.



2.1 сурет – Екілік іздеу ағашының блок-схемасы

3. Программалық реализация

3.1 Қолданылған орта және құралдар

Курстық жоба барысында қолданылған орта мен құралдар:

1.VS Code (Visual Studio Code) - жеңіл, қуатты және кеңінен қолданылатын код редакторы, Python тілінде программалар жазу үшін қолданылды. VS Code өзінің кеңейтімдері мен плагиндері арқылы Python программалау тілін толық қолдайды, бұл бағдарламашыларға кодты жазу, түзету және тестілеу барысында көп мүмкіндік береді. VS Code редакторының негізгі артықшылықтары - қарапайым интерфейс, қателерді автоматты түрде анықтау, кодыңыздың синтаксисін түсіну және оны жақсарту, түрлі кеңейтімдермен Python кодын жазуды жеңілдету. Сонымен қатар, бұл редактор Python кодын орындау үшін қажет барлық құралдарды біріктіруде ыңғайлы, әрі оны пайдалану өте қарапайым.

2.Python – жобаның негізгі программалау тілі ретінде таңдалды. Python тілі өзінің қарапайымдылығы, оқылуы мен жазылуының жеңілдігімен танымал. Сонымен қатар, Python-ның көп функционалды кітапханалары бар, бұл оны деректер құрылымдарын құру, алгоритмдерді іске асыру және графикалық интерфейстерді жасау үшін өте ыңғайлы етеді. Python тілі негізінде екілік іздеу ағашының алгоритмін жүзеге асыру және Tkinter кітапханасы арқылы қолданушы интерфейсін жасау орындалды. Python тілі жобаның тиімді жұмысын қамтамасыз ету үшін таңдалды, себебі ол жылдам дамытуды және тестілеуді жеңілдетеді, сонымен қатар үлкен қауымдастық пен қолдау жүйесіне ие.

3.Tkinter - Python тілінде графикалық қолданушы интерфейстерін құру үшін қолданылған стандартты кітапхана. Tkinter көмегімен қолданушы интерфейсі жасалды, онда қолданушы екілік іздеу ағашына мәндерді енгізіп, олардың симметриялы түрде өту нәтижелерін көре алады. Tkinter кітапханасы Python-ның стандартты бөлімі болғандықтан, қосымша орнатуды талап етпейді. Ол программаның интерфейсін жасау барысында бірнеше терезелер мен батырмалар орнату үшін қажетті құралдарды ұсынады. Tkinter-дің қарапайым әрі интуитивті интерфейсі Python кодын қолданушыға ыңғайлы және түсінікті етіп жасайтын негізгі факторлардың бірі болды.

4.Python стандартты кітапханалары - Python-ның басқа да стандартты кітапханалары қолданылды. Мысалы, tkinter.simpledialog және tkinter.messagebox кітапханалары арқылы қолданушыдан мәліметтерді сұрау және нәтижелерді көрсету жүзеге асырылды. Бұл кітапханалар Python-ның мүмкіндіктерін кеңейтіп, Қорытындылай келе, бұл курстық жоба барысында екілік іздеу ағашын құру және оның жұмысын көрсету үшін қолданушы интерфейсі жасалды. Жоба Python программалау тілі негізінде жүзеге асырылды, ал графикалық интерфейсті жасау үшін Tkinter кітапханасы қолданылды. Жобаның негізгі мақсаты — екілік іздеу ағашының жұмыс істеу принциптерін зерттеу, оны программалау арқылы жүзеге асыру және қолданушыға ыңғайлы түрде көрсету болды. Осы мақсаттарға толық қол жеткізілді.

Екілік іздеу ағашы — деректер құрылымдарының маңызды түрлерінің бірі болып табылады. Ол мәліметтерді тиімді өңдеу, іздеу, қосу және жою операцияларын орындауға мүмкіндік береді. Жобада ағаштың құрылымы Python тілі арқылы алгоритмдік түрде жүзеге асырылды. Ағаштың түйіндерін қосу, симметриялы түрде өту сияқты негізгі функциялар дұрыс әрі тиімді жұмыс істейтіндей етіп жасалды. Бұл ағаш құрылымының артықшылықтары қолданушы енгізген мәліметтерді реттеуде және оларды жылдам көрсету мүмкіндігін қамтамасыз етуде көрініс тапты.

Сонымен қатар, қолданушы интерфейсі жобаның маңызды бөлігі болды. Tkinter кітапханасы арқылы құрастырылған интерфейс программаның қолданушымен өзара әрекетін жеңілдетті. Интерфейстің қарапайым әрі түсінікті болуы пайдаланушыға екілік ағашқа мәндер қосуға және ағаштың симметриялы өту нәтижесін жылдам көруге мүмкіндік берді. Бұл әдіс екілік іздеу ағашының жұмыс принциптерін визуалды түрде түсінуге айтарлықтай көмек көрсетті.

Жобаның практикалық құндылығы жоғары. Ол тек теориялық білімді қолдану емес, сонымен қатар программалау, алгоритмдерді жүзеге асыру және қолданушы интерфейсімен жұмыс істеу дағдыларын дамытуға мүмкіндік берді. Жоба барысында VS Code код редакторы қолданылды, бұл жұмыс істеуді барынша тиімді етті. Python тілінің қарапайымдылығы мен мүмкіндіктері жобаны жылдам әрі сапалы жүзеге асыруға ықпал етті.

Жалпы, бұл курстық жоба екілік іздеу ағашының жұмысын түсіндіру, оны программалық түрде жүзеге асыру және қолданушымен өзара әрекеттесетін интерфейс құру арқылы қойылған мақсаттарды толығымен орындады. Жоба барысында алынған білім мен тәжірибе деректер құрылымдарын зерттеуде және программалауда қолданылатын маңызды дағдыларды нығайтты. Алдағы уақытта бұл жобаны қосымша функциялармен толықтырып, оны деректерді өңдеудің неғұрлым күрделі сценарийлерінде қолдануға болады. визуализациясын жеңілдетеді. Сонымен қатар, бұл құралдар арқылы қолданушыға ыңғайлы интерфейс пен жұмыс істегенде оңай пайдаланылатын функциялар қамтамасыз етілді.

Бұл құралдар мен орта жоба барысында тиімді әрі ыңғайлы жұмыс істеуге мүмкіндік берді. VS Code редакторы мен Python тілінің кең мүмкіндіктері жобаның барлық кезеңдерінде өнімді жұмыс істеуді және тестілеуді жеңілдетті, ал Tkinter кітапханасы қолданушыға ыңғайлы интерфейс жасауға көмектесті.

3.2 Қолданушы интерфейсін жобалау

Бұл курстық жоба екілік іздеу ағашы мен қолданушы интерфейсін жобалауды қамтиды. Екілік іздеу ағашы (Binary Search Tree, BST) - бұл деректер құрылымы, онда әрбір түйіннің сол жақ балалары оның мәнінен кіші, ал оң жақ балалары одан үлкен болады. Мұндай құрылым деректерді тиімді іздеу, қосу, жою сияқты операцияларды орындауға мүмкіндік береді. Бұл жобада екілік іздеу ағашының негізгі операциялары Python тілінде жүзеге асырылған: түйіндерді қосу және ағашты симметриялы түрде өту. Қолданушы интерфейсі Tkinter кітапханасы арқылы жасалған. Tkinter - Python тілінде графикалық интерфейс құруға арналған стандартты кітапхана. Интерфейсте қолданушыға екілік іздеу ағашына мәндер қосуға, сондай-ақ ағашты симметриялы өту арқылы көріп шығуға мүмкіндік беретін батырмалар орналасқан. Интерфейстің негізгі элементтері: “Вставить” батырмасы - қолданушы жаңа мән енгізіп, оны ағашқа қосу үшін пайдаланады; “Показать обход” батырмасы - ағаштың барлық түйіндерін симметриялы өту арқылы көрсету үшін қолданылады. Бұл батырмаларды басу арқылы қолданушы екілік іздеу ағашымен өзара әрекет жасай алады. Әрбір енгізілген мән ағашқа қосылып, нәтижесі экранда көрсетіледі. Қолданушы интерфейсінде мәнді енгізу үшін арнайы терезе ашылады. Осы терезеге енгізілген мән автоматты түрде екілік іздеу ағашына қосылады. Қолданушы ағаштың симметриялы өту нәтижесін көру үшін “Показать обход” батырмасын басады. Бұл кезде ағаштағы барлық түйіндер солдан оңға қарай өтіп, нәтижесі экранда шығарылады. Бұл курстық жоба арқылы екілік іздеу ағашының жұмысын үйренуге және оның графикалық интерфейсте қалай қолданылатынын көруге болады. Tkinter көмегімен жасалған қолданушы интерфейсі екілік ағашпен жұмыс істеуді жеңілдетеді және оның нәтижелерін визуализациялауға мүмкіндік береді.

3.3 Қолданушы интерфейсін тестілеу

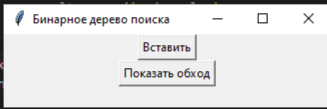
Қолданушы интерфейсінің жұмысын тексеру үшін тестілеудің барлық негізгі және шектік жағдайлары қарастырылды. Тестілеу барысында қолданушының әртүрлі әрекеттерін ескере отырып, Қорытындылай келе, бұл курстық жоба барысында екілік іздеу ағашын құру және оның жұмысын көрсету үшін қолданушы интерфейсі жасалды. Жоба Python программалау тілі негізінде жүзеге асырылды, ал графикалық интерфейсті жасау үшін Tkinter кітапханасы қолданылды. Жобаның негізгі мақсаты — екілік іздеу ағашының жұмыс істеу принциптерін зерттеу, оны программалау арқылы жүзеге асыру және қолданушыға ыңғайлы түрде көрсету болды. Осы мақсаттарға толық қол жеткізілді.

Екілік іздеу ағашы — деректер құрылымдарының маңызды түрлерінің бірі болып табылады. Ол мәліметтерді тиімді өңдеу, іздеу, қосу және жою операцияларын орындауға мүмкіндік береді. Жобада ағаштың құрылымы Python тілі арқылы алгоритмдік түрде жүзеге асырылды. Ағаштың түйіндерін қосу, симметриялы түрде өту сияқты негізгі функциялар дұрыс әрі тиімді жұмыс істейтіндей етіп жасалды. Бұл ағаш құрылымының артықшылықтары қолданушы енгізген мәліметтерді реттеуде және оларды жылдам көрсету мүмкіндігін қамтамасыз етуде көрініс тапты.

Сонымен қатар, қолданушы интерфейсі жобаның маңызды бөлігі болды. Tkinter кітапханасы арқылы құрастырылған интерфейс программаның қолданушымен өзара әрекетін жеңілдетті. Интерфейстің қарапайым әрі түсінікті болуы пайдаланушыға екілік ағашқа мәндер қосуға және ағаштың симметриялы өту нәтижесін жылдам көруге мүмкіндік берді. Бұл әдіс екілік іздеу ағашының жұмыс принциптерін визуалды түрде түсінуге айтарлықтай көмек көрсетті.

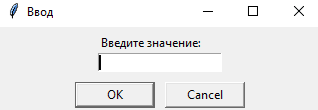
Жобаның практикалық құндылығы жоғары. Ол тек теориялық білімді қолдану емес, сонымен қатар программалау, алгоритмдерді жүзеге асыру және қолданушы интерфейсімен жұмыс істеу дағдыларын дамытуға мүмкіндік берді. Жоба барысында VS Code код редакторы қолданылды, бұл жұмыс істеуді барынша тиімді етті. Python тілінің қарапайымдылығы мен мүмкіндіктері жобаны жылдам әрі сапалы жүзеге асыруға ықпал етті.

Жалпы, бұл курстық жоба екілік іздеу ағашының жұмысын түсіндіру, оны программалық түрде жүзеге асыру және қолданушымен өзара әрекеттесетін интерфейс құру арқылы қойылған мақсаттарды толығымен орындады. Жоба барысында алынған білім мен тәжірибе деректер құрылымдарын зерттеуде және программалауда қолданылатын маңызды дағдыларды нығайтты. Алдағы уақытта бұл жобаны қосымша функциялармен толықтырып, оны деректерді өңдеудің неғұрлым күрделі сценарийлерінде қолдануға болады. жұмысындағы ықтимал ақаулар анықталып, функционалдықтың талаптарға сәйкестігі бағаланды.Енгізу өрісін тексеру бірінші кезекте жүргізілді. Енгізу өрісіне тек сандық мәндерді енгізуге рұқсат беріледі(3.1-сурет).

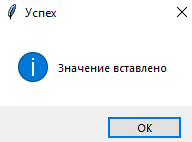


3.1-сурет

Егер қолданушы сан енгізсе, программа оны бинарлық іздеу ағашына қосады (3.2-сурет) және «Значение вставлено» деген хабарлама көрсетеді(3.3-сурет). Бұл әрекет бірнеше мәрте қайталанып, ағаштың құрылымына мәндердің дұрыс қосылғаны тексерілді. Мысалы, 20, 15, 25 сандарын енгізгенде, олар ағашта ретімен сақталды.

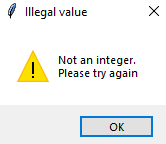


3.2-сурет



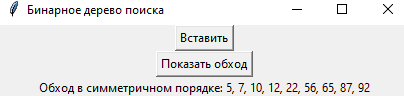
3.3-сурет

Ал егер енгізу өрісіне мәтін, арнайы символдар немесе бос орындар сияқты жарамсыз деректер енгізілсе, программа мұндай мәндерді қабылдамай, қате деген мән шығарады (3.4-сурет).



3.4-сурет

«Вставить» батырмасының жұмысы әртүрлі сценарийлерде егжей-тегжейлі тексерілді. Сан енгізген кезде батырма мәнді ағашқа сәтті қосқанын көрсетті. Қайталанатын мәндерді енгізу әрекеттері кезінде Қорытындылай келе, бұл курстық жоба барысында екілік іздеу ағашын құру және оның жұмысын көрсету үшін қолданушы интерфейсі жасалды. Жоба Python программалау тілі негізінде жүзеге асырылды, ал графикалық интерфейсті жасау үшін Tkinter кітапханасы қолданылды. Жобаның негізгі мақсаты — екілік іздеу ағашының жұмыс істеу принциптерін зерттеу, оны программалау арқылы жүзеге асыру және қолданушыға ыңғайлы түрде көрсету болды. Осы мақсаттарға толық қол жеткізілді.( 3.5-сурет)



3.5-сурет

Қорытынды

Бұл курстық жоба барысында екілік іздеу ағашын құру және оның жұмысын көрсету үшін қолданушы интерфейсі жасалды. Жоба Python программалау тілі негізінде жүзеге асырылды, ал графикалық интерфейсті жасау үшін Tkinter кітапханасы қолданылды. Жобаның негізгі мақсаты — екілік іздеу ағашының жұмыс істеу принциптерін зерттеу, оны программалау арқылы жүзеге асыру және қолданушыға ыңғайлы түрде көрсету болды. Осы мақсаттарға толық қол жеткізілді.

Екілік іздеу ағашы — деректер құрылымдарының маңызды түрлерінің бірі болып табылады. Ол мәліметтерді тиімді өңдеу, іздеу, қосу және жою операцияларын орындауға мүмкіндік береді. Жобада ағаштың құрылымы Python тілі арқылы алгоритмдік түрде жүзеге асырылды. Ағаштың түйіндерін қосу, симметриялы түрде өту сияқты негізгі функциялар дұрыс әрі тиімді жұмыс істейтіндей етіп жасалды. Бұл ағаш құрылымының артықшылықтары қолданушы енгізген мәліметтерді реттеуде және оларды жылдам көрсету мүмкіндігін қамтамасыз етуде көрініс тапты.

Сонымен қатар, қолданушы интерфейсі жобаның маңызды бөлігі болды. Tkinter кітапханасы арқылы құрастырылған интерфейс программаның қолданушымен өзара әрекетін жеңілдетті. Интерфейстің қарапайым әрі түсінікті болуы пайдаланушыға екілік ағашқа мәндер қосуға және ағаштың симметриялы өту нәтижесін жылдам көруге мүмкіндік берді. Бұл әдіс екілік іздеу ағашының жұмыс принциптерін визуалды түрде түсінуге айтарлықтай көмек көрсетті.

Жобаның практикалық құндылығы жоғары. Ол тек теориялық білімді қолдану емес, сонымен қатар программалау, алгоритмдерді жүзеге асыру және қолданушы интерфейсімен жұмыс істеу дағдыларын дамытуға мүмкіндік берді. Жоба барысында VS Code код редакторы қолданылды, бұл жұмыс істеуді барынша тиімді етті. Python тілінің қарапайымдылығы мен мүмкіндіктері жобаны жылдам әрі сапалы жүзеге асыруға ықпал етті.

Жалпы, бұл курстық жоба екілік іздеу ағашының жұмысын түсіндіру, оны программалық түрде жүзеге асыру және қолданушымен өзара әрекеттесетін интерфейс құру арқылы қойылған мақсаттарды толығымен орындады. Жоба барысында алынған білім мен тәжірибе деректер құрылымдарын зерттеуде және программалауда қолданылатын маңызды дағдыларды нығайтты. Алдағы уақытта бұл жобаны қосымша функциялармен толықтырып, оны деректерді өңдеудің неғұрлым күрделі сценарийлерінде қолдануға болады.

Пайдаланылған әдебиеттер тізімі

1. Алексеев В.В. – Алгоритм теориясы. Оқу-әдістемелік құрал. SarFTI NRNU MEPhI, 2021 ж

2. Бағдарламалаудың негізгі құралдары / В.Н. Шакин – М.: Форум: SRC INFRA-M, 2019. – 303 б.

3. Балдин, К. В. Математикалық бағдарламалау: Оқу құралы / К. В. Балдин, Н. А. Брызгалов, А. В. Рукосуев; Жалпы ред. Балдина К.В. – 2-ші басылым. – Мәскеу: Баспа-сауда корпорациясы «Дашков и К» ЖШҚ, 2018. – 218 б.

4. Екілік ағаштар [Электрондық ресурс] – кіру режимі: http://www.codenet.ru/progr/alg/btree.php

5. Борозный, А.А. Бинарлы ағашта іздеудің патрициандық алгоритмін оңтайландыру / А.А. Борозный // Гагарин оқулары - 2019: XLV Халықаралық жастар ғылыми конференциясының тезистер жинағы, Мәскеу, Барнаул, Ахтубинск, 16-19 сәуір / 2019 ж. Мәскеу авиация институты (ұлттық зерттеу университеті). – Мәскеу, Барнаул, Ахтубинск: Мәскеу авиация институты (Ұлттық зерттеу университеті), 2019. – 724-б.

6. Голицына О.Л. Бағдарламалау тілдері: Оқу құралы / О.Л. Голицына, Т.Л. Partyka, I.I. Попов. – 3-бас., қайта қаралған. және қосымша – М.: Форум: INFRA-M, 2021. – 399 б.

7. Бағдарламашыларға арналған дискретті математика / Гари Хаггард, Джон Шлипф, Сью Уайтсайдс - М.: Бином. Білім зертханасы, 2018. – 632 б.

8. Долгушин, Д.А. Екілік ағаштарды кестені іздеу есептеріне қолдану / Д.А. Долгушин // Нижневартовск мемлекеттік университетінің ХХ Бүкілресейлік студенттік ғылыми-практикалық конференциясы: мақалалар жинағы, Нижневартовск, 03–04 сәуір 2018 ж. / Жауапты редактор А.ИН. Коричко. – Нижневартовск: Нижневартовск мемлекеттік университеті, 2018. – С.182-184.

9. Информатика. IT мамандықтарының студенттеріне арналған кіріспе курс: оқу құралы / В.К. – Қорған: Қорған мемлекеттік баспасы. Университет, 2020. – 218 б.

10. Калинина, Е.А. Алгоритм теориясы: оқу-әдістемелік кешенін жасау және қолдану / Е.А. Калинина // Педагогикалық информатика. – 2019. – No 1. – 53-58 Б.

11. Лебеденко, Л.Ф.Информатика: оқу-әдістемелік құрал / Л.Ф.Лебеденко, Т.И.Парначева. – Новосибирск: Сібір мемлекеттік байланыс және информатика университеті, 2019. – 137 б.

12. Никитин, А.В. Иерархиялық деректер құрылымдары: екілік ағаштар: оқу құралы / А.В. Никитин, Т.Н.Ничушкина. - Мәскеу: ММУ баспасы им. Н.Е.Бауман, 2018. - 54 б.

13. Пестова И. Динамикалық деректер құрылымдары және деректерді тізімдік құрылымдарға ұйымдастыру // «Анықтамалық» оқу порталы. — Мақала жазылған күні:01.03.2021ж.—URL

https://spravochnick.ru/informatika/dinamicheskie\_struktury\_dannyh\_i\_organizaciya\_dannyh\_v\_spiskovye\_struktury

14. Компьютерлік бағдарламаны мемлекеттік тіркеу туралы куәлік № 2018618935 Ресей Федерациясы. Адаптивті жады бар популяциялық кездейсоқ іздеуге негізделген екілік іздеу кеңістігіндегі оңтайландыру бағдарламасы: № 2018616268: қолданба. 18.06.2018: басылым. 24.07.2018 / Қ.С.Сарин, И.А.Ходашинский; өтініш беруші «Томск мемлекеттік басқару жүйелері және радиоэлектроника университеті» Федералдық мемлекеттік бюджеттік жоғары оқу орны (TUSUR).

15. Сейдаметова, З.С. Теңгерімді екілік іздеу ағаштарын зерттеудің ерекшеліктері / З.С.Сейдаметова // Экономикадағы, білім берудегі және әлеуметтік саладағы ақпараттық және компьютерлік технологиялар. – 2019. – No 3(25). – 83-93 б.

16. Қазіргі заманғы бағдарламалау: I Халықаралық ғылыми-практикалық конференция материалдары (Нижневартовск, 15–18 қараша 2018 ж.) / реп. ред. Т.Б. Қазиахмедов. – Нижневартовск: Нижневартовск мемлекеттік университеті, 2018 – 200 б.

17. Деректер құрылымдары - ағаштар - [Электрондық ресурс] - кіру режимі: https://pro-prof.com/archives/682

Қосымша А Программалық код

class Node:

def \_\_init\_\_(self, key):

self.left = None

self.right = None

self.value = key

class BinarySearchTree:

def \_\_init\_\_(self):

self.root = None

def insert(self, key):

if self.root is None:

self.root = Node(key)

else:

self.\_insert\_recursively(self.root, key)

def \_insert\_recursively(self, current\_node, key):

if key < current\_node.value:

if current\_node.left is None:

current\_node.left = Node(key)

else:

self.\_insert\_recursively(current\_node.left, key)

elif key > current\_node.value:

if current\_node.right is None:

current\_node.right = Node(key)

else:

self.\_insert\_recursively(current\_node.right, key)

def inorder\_traversal(self, node, result):

if node:

self.inorder\_traversal(node.left, result)

result.append(node.value)

self.inorder\_traversal(node.right, result)

Tkinter көмегімен интерфейс құру

Әрі қарай, Tkinter көмегімен GUI жасаймыз, ол бізге мәндерді енгізуге және симметриялы тәртіпте ағаштың өтуін көрсетуге мүмкіндік береді.

import tkinter as tk

from tkinter import simpledialog, messagebox

class TreeApp:

def \_\_init\_\_(self, master):

self.master = master

master.title("Бинарное дерево поиска")

self.bst = BinarySearchTree()

self.insert\_button = tk.Button(master, text="Вставить", command=self.insert\_value)

self.insert\_button.pack()

self.show\_button = tk.Button(master, text="Показать обход", command=self.show\_inorder)

self.show\_button.pack()

self.result\_label = tk.Label(master, text="")

self.result\_label.pack()

def insert\_value(self):

value = simpledialog.askinteger("Ввод", "Введите значение:")

if value is not None:

self.bst.insert(value)

messagebox.showinfo("Успех", "Значение вставлено")

def show\_inorder(self):

result = []

self.bst.inorder\_traversal(self.bst.root, result)

self.result\_label.config(text="Обход в симметричном порядке: " + ", ".join(map(str, result)))

if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

app = TreeApp(root)

root.mainloop()