**Algorytmy zagadnienia**

**1. Na przykładzie wektora 5 – 1 – 7 – 4– 3 wyjaśnij działanie algorytmu sortowania przez proste wstawianie. Kiedy lepszy jest algorytm sortowania przez proste wybieranie a kiedy przez proste wstawianie?**

**Przebieg działania:**

[5,1,7,4,3]🡪[1,5,7,4,3]🡪[1,4,5,7,3]🡪[1,3,4,5,7]

Opis działania:

* Na samym początku bierzemy dowolny element z nieposortowanego wektora.
* Następnie wyciągnięty element zaczynamy porównywać z kolejnymi elementami zbioru posortowanego, póki nie napotkamy elementu równego lub elementu większego (jeśli chcemy otrzymać ciąg niemalejący) lub nie znajdziemy się na początku/końcu zbioru uporządkowanego.
* Nasz wyciągnięty element wstawiamy w miejsce, tam gdzie skończyliśmy porównywać.
* Jeżeli zbiór elementów nieuporządkowanych jest niepusty, zaczynamy kroki od początku wybierając inny element.

**Linki:**

**2. Na przykładzie wektora 5 – 1 – 7 – 4– 3 wyjaśnij działanie algorytmu sortowania przez proste wybieranie. Kiedy lepszy jest algorytm sortowania przez proste wybieranie a kiedy przez proste wstawianie?**

**Przebieg działania:**

[5,1,7,4,3]🡪[1,5,7,4,3]🡪[1,3,5,7,4]🡪[1,3,4,5,7]🡪[1,3,4,5,7]🡪

[1,3,4,5,7]

Opis działania:

* Na samym początku zaczynamy wyszukiwać minimalną wartość z tablicy spośród elementów od *i* do końca tablicy.
* Kolejnym etapem jest zamieniamy wartości minimalnej, z elementem na pozycji *i*.

**Linki:**

**3. Na przykładzie wektora 5 – 1 – 7 – 4– 3 wyjaśnij działanie algorytmu sortowania bąbelkowego. Co to jest stabilność sortowania?**

**Przebieg działania:**

[5,1,7,4,3] 🡪[1,5,7,4,3]🡪[1,5,7,4,3]🡪[1,5,4,7,3]🡪[1,5,4,3,7]🡪[1,5,4,3,7]🡪

[1,4,5,3,7]🡪[1,4,3,5,7]🡪[1,4,3,5,7]🡪[1,4,3,5,7]🡪[1,3,4,5,7]🡪

[1,3,4,5,7]🡪[1,3,4,5,7]🡪[1,3,4,5,7]

Opis działania:

* Na samym początku porównujemy ze sobą kolejne elementy.
* Jeśli element n+1 jest większy od n, wtedy zamieniamy je miejscami i przechodzimy do następnych dwóch elementów.
* Gdy dojdziemy do końca wektora, wracamy na sam początek.
* Czynność tak naprawdę będziemy powtarzać dopóki wszystkie elementy nie będą posortowane.

**Linki:**

**4. Wyjaśnij słowami zasadę działania algorytmu szybkiego sortowania**

Algorytm szybkiego sortowania jest algorytmem rekurencyjnym. Jest on tak naprawdę jednym z najszybszych algorytmów sortujących dane, które stworzono/wynaleziono. Jest on chętnie implementowany i wdrażany do systemów informatycznych ze względu na szerokie spektrum. Jego złożoność wynosi dla większości przypadków *O(nlogn)*, lecz jego złożoność pesymistyczna wynosi *O(n2)*, są to bardziej przypadki bardzo skrajne i z reguły rzadko występują.

Sam algorytm możemy krótko scharakteryzować w **trzech punktach**:

1. **Dziel** – tutaj problem główny zostaje podzielony na podproblemy.
2. **Zwyciężaj** – tutaj znajdujemy rozwiązanie naszych podproblemów.
3. **Połącz** – rozwiązania podproblemów zostają połączone w rozwiązanie problemu głównego.

**Działanie algorytmu szybkiego sortowania:**

Algorytm ten wykorzystuje technikę "dziel i zwyciężaj". Według ustalonego schematu wybierany jest jeden element w sortowanej tablicy, który będziemy nazywać **pivot.** Pivot może być tak naprawdę elementem środkowym, pierwszym, ostatnim, losowym lub wybranym według jakiegoś innego schematu dostosowanego do zbioru danych. Następnie ustawiamy elementy nie większe na lewo tej wartości, natomiast nie mniejsze na prawo. W ten sposób powstaną nam dwie części tablicy (niekoniecznie równe), gdzie w pierwszej części znajdują się elementy nie większe od drugiej. Następnie każdą z tych podtablic sortujemy osobno według tego samego schematu.

**Linki:**

* <http://www.algorytm.edu.pl/algorytmy-maturalne/quick-sort.html>
* <https://eduinf.waw.pl/inf/alg/003_sort/0018.php>
* <https://mattomatti.com/pl/fs07>

**5. Algorytm a jego implementacja.**

Komputery przetwarzają przekazywane im informacje z wykorzystaniem algorytmów. Program jest algorytmem zapisanym w języku zrozumiałym dla maszyny (kodzie maszynowym). Każdy poprawny kod maszynowy da się przełożyć na zestaw instrukcji dla teoretycznego modelu komputera – maszyny Turinga.

Zwykle algorytmy pracują na danych wejściowych i uzyskują z nich dane wyjściowe. Informacje zapisane w pamięci maszyny traktuje się jako jej stan wewnętrzny. Niektóre algorytmy mają za zadanie wyłącznie przeprowadzanie komputera z jednego stanu wewnętrznego do innego.

Każdy algorytm komputerowy musi być wprowadzony do komputera w bardzo rygorystycznie zdefiniowanym języku. Ludzie często komunikując się, przesyłają między sobą informacje wieloznaczne. Komputery mogą reagować tylko na całkowicie jednoznaczne instrukcje. Jeżeli dany algorytm da się wykonać na maszynie o dostępnej mocy obliczeniowej i pamięci oraz w akceptowalnym czasie, to mówi się, że jest obliczalny.

Poprawne działanie większości algorytmów implementowanych w komputerach opiera się na kolejnej realizacji pewnego zestawu warunków. Jeżeli któryś z nich nie zostanie spełniony, to program kończy się komunikatem o błędzie. Czasami podczas implementacji algorytmu nawet istotny warunek może zostać pominięty. Przykładowo, w programie dzielącym przez siebie dwie liczby użytkownik poleca wykonać dzielenie przez zero. Działanie aplikacji, która nie sprawdzi warunku „czy dzielnik nierówny zero”, zostanie przerwane przez system operacyjny komputera.

**Linki:**

* <https://pl.wikipedia.org/wiki/Algorytm#Implementacja_algorytm%C3%B3w>

**6. Wymień i krótko scharakteryzuj podstawowe i dynamiczne struktury danych.**

**Struktury podstawowe** tak naprawdę nie zmieniają obszaru pamięci w trakcie realizacji programu.

**Podstawowe struktury danych**:

* **Tablica** - jest to struktura danych przechowująca sekwencję wartości tego samego typu, która obsługuje tworzenie, dostęp indeksowy, przypisanie indeksowe oraz iteracje.
* **Zbiór** – jest to kolekcja elementów, która nie może zawierać duplikatów.
* **Rekord** - jest on jedną z prostszych struktur danych, najbardziej nadaje się do prezentowania informacji o jednym obiekcie. Dlaczego? Dlatego, że w trakcie działania algorytmu nie zmienia on swego rozmiaru oraz struktury.
* **Plik sekwencyjny** – jest to plik, który nasz komputer może czytać tylko od początku do końca, a zapisywać poprzez dopisanie kolejnej porcji danych na końcu.

**Dynamiczne struktury danych** – jest to tak naprawdę metoda przechowywania danych przez program. Jest to bardzo ważne pojęcie z dziedziny programowania. Na czym polegają dynamiczne struktury danych? Głównie polegają one na tym, że dane są przechowywane w rezerwowanych na bieżąco miejscach pamięci, które są powiązane ze sobą przy użyciu wskaźników. Głównie w odróżnieniu od klasycznych zmiennych czy też tablic, dostęp do danych odbywa się tak naprawdę w sposób pośredni, za pomocą wskaźników zawartych w każdym elemencie struktury.

**Struktury dynamiczne:**

* **Stos** – jest nazywany również kolejką LIFO, czyli „ostatni na wejściu, pierwszy na wyjściu”. Często jest przydatny w sytuacjach, gdy chcemy przeprowadzić operację najpierw na najnowszych danych, a dopiero później na starszych.
* **Kolejka** – jest ona tak naprawdę przeciwieństwem stosu, nazywana jest inaczej kolejką FIFO, czyli „pierwszy na wejściu, pierwszy na wejściu”. Tutaj wszystko odbywa się inaczej, zaczynamy tak naprawdę przetwarzanie danych od elementów, które pojawiły się jako pierwsze.
* **Lista jednokierunkowa** – w niej tak naprawdę każdy element zawiera referencję do kolejnego (lub poprzedniego) komponentu w liście.
* **Lista dwukierunkowa** – jest ona jakby rozwinięciem listy jednokierunkowej.
* **Drzewo** -
* **Sterty** – jest to struktura danych oparta na drzewie. W niej wartości potomków węzła są w stałej relacji z wartością rodzica.

**Linki:**

* <https://informatyka.fandom.com/wiki/Dynamiczne_struktury_danych>
* <https://pl.wikipedia.org/wiki/Kopiec_(informatyka)>
* <https://pl.wikipedia.org/wiki/Plik_sekwencyjny>

**7. Podaj definicję tablicy. W jaki sposób tablice są tworzone, jak uzyskuje się dostęp do ich elementów?**

**Tablica** – jest to struktura danych przechowująca sekwencję wartości tego samego typu, która obsługuje tworzenie, dostęp indeksowy, przypisanie indeksowe oraz iteracje.

**W jaki sposób są tworzone?**

Tablice są tworzone w **3 etapach**. Są to min.:

* **Zadeklarowanie tablicy** – określamy jej nazwę oraz typ danych, jaki będzie przechowywać.
* **Utworzenie tablicy** – określamy jej wielkość (liczbę elementów).
* **Zainicjalizowanie elementów tablicy** – przypisujemy wartość do każdego elementu.

**Przykład tworzenia tablicy (język Java):**

String[] daysOfWeek = new String[7];

daysOfWeek[0] = "poniedziałek";

daysOfWeek[1] = "wtorek";

daysOfWeek[2] = "środa";

daysOfWeek[3] = "czwartek";

daysOfWeek[4] = "piątek";

daysOfWeek[5] = "sobota";

daysOfWeek[6] = "niedziela";

**Opis:**

Powyższy przykład pokazuje jak możemy stworzyć tablicę dla kilku elementów w języku Java. Pierwszym etapem jest tak naprawdę stworzenie naszej tablicy (w przykładzie posiada ona nazwę **daysOfWeek**) obiektów typy String (będziemy mogli w niej przechowywać tylko łańcuchy tekstowe). Następnie podaliśmy maksymalną ilość elementów, które nasza tablica będzie przechowywać (maksymalna ilość wynosi 7). Następnie dodajemy elementy do naszej tablicy, są one numerowane od 0 do 6 w tym przypadku.

**Jak uzyskuje się dostęp do ich elementów?**

Tak naprawdę dostęp do elementu w danej tablicy jest realizowany poprzez podanie jego pozycji (indeksów) w nawiasach kwadratowych.

**Przykład:**

np. tab[5], t[2][3]

**Linki:**

* <https://www.samouczekprogramisty.pl/tablice-w-jezyku-java/>

**8. Co to jest plik? Jakie są elementarne operacje na plikach?**

**Plik** – jest to tak naprawdę uporządkowany zbiór danych o skończonej długości, który posiada szereg atrybutów oraz stanowi dla użytkownika systemu operacyjnego jednolitą całość. Pliki możemy zapisywać na komputerze jak i na płytach czy na dyskach pamięci typu pen- drivie. Nie mamy tak naprawdę ograniczeń jeżeli chodzi o wielkość plików oraz ich zawartości. Muszą być one przechowywane w odpowiedni sposób, tak by nikt nieodpowiedni nie miał dostępu do zapisanych przez nas informacji. Obecnie pliki można przesyłać sobie za pomocą Internetu dzięki czemu trafiają do właściwego odbiorcy. Wszystkie pliki należy zachować gdyż mogą okazać się, że będziemy chcieli jeszcze z nich skorzystać i nie będziemy musieli tworzyć nowych.

Jakie są typy plików?

* **Katalogi** – są to tak naprawdę pliki zawierające spis odwołań do innych plików.
* **Pliki wykonywalne** – zawierają one głównie program, który musi być wykonany lub polecenia dla interpretera.
* **Kolejki FIFO** – są to pliki, które realizują bardziej skomplikowane zadania.
* **Dowiązania symboliczne** – inaczej mówiąc odwołanie do innego pliku.

Możemy tak naprawdę wyróżnić jeszcze kilka typów plików w **systemie DOS**:

* **Pliki tekstowe** – są to dane zapisane w formie kodowania ASCII łącznie z kodami sterującymi urządzeniami.
* **Pliki binarne** – są to inaczej pozostałe pliki.

**Jakie są elementarne operacje na plikach?**

* **Tworzenie pliku** – wymaga on jawnego podania wartości niektórych atrybutów.
* **Zapis** –konieczne jest określenie, co ma być zapisane i gdzie ma być zapisane.
* **Odczyt** – konieczne jest określenie, co ma być odczytane i gdzie mają być umieszczone te dane.
* **Pozycjonowanie (seek)** – inaczej przesuwanie kursora.
* **Usuwanie pliku** – należy określić plik do usunięcia. Usuwane są min. zawartość oraz wpis ewidencyjny pliku.
* **Zerowanie pliku** (zapis od zera).

**Dodatkowe operacje:**

* Dopisywanie na końcu pliku.
* Otwieranie.
* Zamykanie.
* Zmiana nazwy pliku.

**Linki:**

* <http://www.zpcir.ict.pwr.wroc.pl/~witold/opsys/os_fsys_h.pdf>
* <https://pl.wikipedia.org/wiki/Plik_danych>
* <http://smurf.mimuw.edu.pl/node/926>

**9. Struktury plików; sposoby wyróżniania pól.**

**10. Co to jest rekord? Porównaj rekord z tablicą.**

**Rekord** – jest to złożony typ danych, który występuje w wielu językach programowania. Jego zadaniem jest grupowanie logicznie powiązanych ze sobą danych różnego typu w jednym obszarze pamięci. Składowe struktury czyli pola, są inaczej mówiąc etykietowane, tj. posiadają one swoje unikatowe nazwy, głównie poprzez podanie nazwy otrzymuje się dostęp do danego pola.

**Porównanie rekordu z tablicą:**

Jeżeli chodzi o porównanie tablicy z rekordem to uważam, że tablica tak naprawdę służy do przechowywania dużej ilości danych tego samego typu. Tablica jest jakby pudełkiem, które jest podzielona na komórki. W każdej komórce możemy przechowywać daną wartość (liczbę, tekst, itp.).

Lecz jeżeli chodzi o rekordy to tak naprawdę służą one do przechowywania danych, które są powiązane ze sobą, ale są one różnych typów. Tak naprawdę tablica i rekord jest podobna do siebie, lecz różnią się one od siebie pod względem przechowywanych danych. Inaczej mówiąc w tablicy nie możemy przechowywać elementów różnych typów. Przechowywanie różnych typów może nam umożliwić rekord.

**Linki:**

* <https://pl.wikipedia.org/wiki/Rekord_(informatyka)>
* <http://nanophysics.pl/teaching/metody_programowania/metody_programowania05.pdf>
* <https://zsobobowa.eu/pliki/program/pas10.pdf>

**11. Listowe i tablicowe implementacje stosów, zalety i wady.**

**Listowa implementacja stosów:**

Do realizacji stosu możemy wykorzystać tak naprawdę listę jednokierunkową. Zapis na stos będzie polegać głównie na umieszczaniu elementu na początku listy. Szczyt stosu będzie pierwszym elementem listy. Odczyt ze stosu będzie równoważny odczytowi pierwszego elementu listy, lecz usunięcie ze stosu będzie odpowiadało usunięciu elementu z początku listy. Realizacja listowa jest szczególnie wygodna wtedy, gdy nie znamy maksymalnego rozmiaru stosu – w przeciwieństwie do tablic listy mogą swobodnie rosnąć w pamięci, dopóki jest dla nich miejsce. W podanych niżej procedurach nie obsługujemy sytuacji braku pamięci – w każdym ze środowisk programowania można w takim przypadku wykorzystać mechanizmy wyłapywania błędów, które jednakże zaciemniają realizowane funkcje.

**Tablicowa implementacja stosów:**

Do utworzenia stosu w tablicy potrzebujemy dwóch zmiennych. Pierwszą z nich będzie *tablica*, która przechowuje umieszczone na stosie elementy. Druga zmienna to *sptr* - będzie ona służyć do zapamiętywania pozycji szczytu stosu i nosi nazwę wskaźnika stosu. Przyjmujemy, że wskaźnik stosu zawsze wskazuje pustą komórkę tablicy, która znajduje się tuż ponad szczytem stosu: Po utworzeniu tablicy zmienna sptr musi zawsze być zerowana. Stos jest pusty, gdy sptr wskazuje początek tablicy, czyli komórkę o indeksie zero. Ta własność jest wykorzystywana w operacji **empty**. Stos jest pełny, gdy sptr ma wartość równą liczbie komórek tablicy. W takim przypadku na stosie nie można już umieszczać żadnych dalszych danych. gdyż trafiłyby poza obszar zarezerwowany na tablicę

**Wady i zalety tablicowej implementacji stosów:**

* **Wady**
* Sposób jest tak naprawdę szybki oraz efektywny, ale posiada negatywną cechę, opiera się głównie na tablicy o stałej długości, w ten sposób ogranicza maksymalną wielkość stosu.
* Możliwość przepełnienia.
* **Zalety**
  + Łatwa implementacja.

**Wady i zalety listowej implementacji stosów:**

* **Wady**
* Trudniejsza implementacja w porównaniu do stosu opartego o tablicę.
* **Zalety**
  + Szybkość działania.
  + Możliwość przechowywania praktycznie nie ograniczonej ilości danych.

**Linki:**

* <https://4programmers.net/Forum/Download/21714>
* <http://strefakodera.pl/algorytmy/abstrakcyjne-struktury-danych/abstrakcyjne-struktury-danych-stos>

**12.** **Co to jest lista? Jakie są elementarne operacje zdefiniowane dla list?**

**Lista** – jest to struktura danych, która służy do reprezentacji zbiorów dynamicznych, w której elementy są ułożone w liniowym porządku. Oczywiście lista podobnie jak i tablica jest przeznaczone do przechowywania większej ilości danych, choć jest jedna różnica, która jest dość znaczna, lista umożliwia tak naprawdę łatwe dodawanie nowych elementów.

**Podsumowanie**:

Listę tak naprawdę możemy postrzegać jako tzw. łańcuszek połączonych ze sobą danych, przy tym wiedząc gdzie znajduje się początek, wtedy możemy znaleźć kolejne ogniwa takiej struktury.

**Elementarne operacje zdefiniowane dla list:**

* **CREATE()** – tworzy nową listę.
* **INSERT()** – dodaje na końcu listy rekord.
* **DELETE()** – usuwa w kolejności rekord z listy. Usunięcie pierwszego i ostatniego rekordu można wykonać w czasie stałym. Dla pozostałych będzie to czas liniowy.
* **FIND()** – pobiera w kolejności rekord z listy.
* **RESET()** – ustawia kursor na podanym rekordzie listy.
* **NEXT()** – zwraca rekord, na którym aktualnie ustawiony jest kursor, po czym przesuwa się na kolejny element listy.
* **PREV()** – jest to analogiczna operacja do NEXT() dla list dwukierunkowych, która umożliwia przesuwanie się w tył.
* **MAKENULL()** – czyści listę, usuwając wszystkie rekordy.

**Linki:**

* <https://stormit.pl/struktury-danych/>
* <http://informatyka1lo.ugu.pl/lista.html>

**13. Wyjaśnij pojęcie stosu, kolejki i pierścienia. Co to są kolejki priorytetowe? Typowe implementacje stosów i kolejek. Problem Józefa.**

**Stos** – jest to tak naprawdę struktura liniowo uporządkowanych danych, z których jedynie ostatni element, zwany wierzchołkiem, jest w danym momencie dostępny. Głównie w wierzchołku odbywa się dołączanie nowych elementów, również jedynie wierzchołek można usunąć. Stos jest tak naprawdę często wykorzystywaną strukturą danych. Działanie na nim jest zazwyczaj porównywane do stosu talerzy: nie można usunąć talerza znajdującego się na dnie stosu nie usuwając wcześniej wszystkich innych. Nie można także dodać nowego talerza w inne miejsce, niż na samą górę.

**Kolejka** – jest ona pokrewna stosowi, lecz jej elementy są dodawane i usuwane z obu stron zgodnie z regułą **first-in, first-out (FIFO)**. Kolejkę opuszcza element, który jest w niej najdłużej.

**Pierścień** – jest to lista, w której ostatni element wskazuje na pierwszy.

**Co to są kolejki priorytetowe**:

**Kolejka priorytetowa** – jest to abstrakcyjny typ danych, który służy do reprezentowania zbioru elementów, z których każdy ma przyporządkowaną wartość zwaną kluczem.

**Zastosowania kolejek priorytetowych:**

Są stosowane w systemach komputerowych, przykładowo w systemach wielozadaniowych programy o najwyższych priorytetach powinny być uruchamiane jako pierwsze.

**Problem Józefa**:

**Problem Józefa Flawiusza** – jest to problem teoretyczny z zakresu kombinatoryki, jest często rozważany w informatyce. W ogólnej wersji problem brzmi następująco: na okręgu ustawiamy n obiektów, następnie eliminujemy co k-ty obiekt, tak długo, aż zostanie tylko jeden. Należy wskazać obiekt, który pozostanie. Dla k = 2 istnieje wzór jawny, a dla pozostałych k istnieją algorytmy rozwiązujące problem między innymi w złożonościach czasowych O(n) i O(k log n).

**Historia:**

Jakby mówiąc inaczej nazwa powyższego problemu nawiązuje do postaci Józefa Flawiusza, który w trakcie wojny rzymsko-żydowskiej został, wraz z grupą 40 żydowskich powstańców otoczony przez Rzymian w jaskini. Powstańcy od pojmania tak naprawdę woleli samobójstwo, dlatego też zdecydowali się utworzyć krąg i zabijać co trzecią osobę, aż zostanie tylko jedna, która popełni samobójstwo. Sam Józef Flawiusz wyliczył, w którym miejscu miał stanąć, aby uniknąć śmierci.

**Linki:**

* <https://eduinf.waw.pl/inf/alg/001_search/0100a.php>
* <https://pl.wikipedia.org/wiki/Stos_(informatyka)>
* <https://pl.wikipedia.org/wiki/Problem_J%C3%B3zefa_Flawiusza>

**14. Wyjaśnij pojęcie drzewa. Co to są drzewa zrównoważone i binarne.**

**Drzewo** – jest to struktura danych, która reprezentuje drzewo matematyczne. W inny sposób reprezentuje ono hierarchię danych (obiektów fizycznych i abstrakcyjnych). Drzewa ułatwiają i przyśpieszają wyszukiwanie, ale także pozwalają w łatwy sposób operować na posortowanych danych.

**Dla każdego drzewa można określić**:

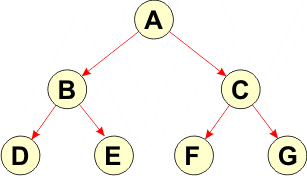
* **Długość drogi** *u* (głębokość)– jest to liczba wierzchołków, przez które należy przejść od korzenia do wierzchołka u.
* **Wysokość** *u*– jest to maksymalna liczba wierzchołków na drodze od u do pewnego wyjścia.
* **Wysokość drzewa** – jest ona równa głębokości z czego sama głębokość jest równa wartości wysokości drzewa, zwiększonej o jeden.
* **Droga** – jest to ścieżka skierowana.
* **Stopień wierzchołka** – jest to liczba jego bezpośrednich następników.
* **Stopień drzewa** – jest to maksymalny stopień wierzchołka.
* **Ścieżka z** *u do v* – zbiór wierzchołków, przez które należy przejść z wierzchołka u do v

**Drzewo zrównoważone:**

**Drzewo zrównoważone** – jest to drzewo, w którym głębokość dowolnych dwóch liści różni się co najwyżej o jeden. Drzewo AVL pozostaje drzewem binarnych poszukiwań. Co może to oznaczać? Tak naprawdę oznacza to, że wierzchołki są uporządkowane w określony sposób. Często przyjmuje się, że elementy w lewym poddrzewie są mniejsze od wierzchołka, lecz w prawym – większe. Zrównoważenie drzewa osiąga się, poprzez przypisanie każdemu węzłowi **współczynnika wyważenia**, który jest równy różnicy wysokości lewego i prawego poddrzewa. Może wynosić on 0, +1 lub -1. Wstawiając lub usuwając elementy drzewa, modyfikuje się także współczynnik wyważenia, a gdy przyjmie on niedozwoloną wartość, wykonuje specjalną operację tzw. rotacji węzłów, która przywraca zrównoważenie.

**Drzewo binarne**:

**Drzewo binarne** – jest to drzewo, w którym węzły mogą posiadać co najwyżej dwóch synów, nazywa się **drzewem binarnym**. Węzły potomne nazywamy odpowiednio **synem lewym** (left child node) i **synem prawym** (right child node). Drzewa binarne mają ogromne znaczenie w informatyce, ponieważ za ich pomocą można odwzorować również drzewa, których węzły posiadają dowolną liczbę synów.

A – korzeń, ojciec B i C  
B,C – synowie A, B – lewy syn A, C – prawy syn A  
D,E – liście, synowie B

F,G – liście, synowie C

**Linki:**

* <http://home.agh.edu.pl/~horzyk/lectures/wdi/WDI-Drzewa.pdf>
* <http://www.algorytm.org/klasyczne/drzewo.html>
* <https://pl.wikipedia.org/wiki/Drzewo_AVL>

**15. Na czym polega szukanie binarne?**

**Szukanie binarne:**

Algorytm wyszukiwania binarnego jest bardzo efektywny, zaliczamy go głównie do jednego z najpopularniejszych algorytmów, który służy tak naprawdę do wyszukiwania danych w tablicy posortowanych liczb. Zatem powyższy algorytm oparty został o metodę zwaną: **„Dziel i zwyciężaj”**, dzięki której jest w stanie w czasie logarytmicznym odnaleźć szukaną liczbę i zwrócić jej indeks (numer indeksu), jeżeli ją odnalazł.

Przykład:

Mamy tablicę zawierającą przykładowo 20 mln elementów. Nasz algorytm, zwany inaczej wyszukiwaniem binarnym będzie musiał wykonać maksymalnie 20 porównań aby odnaleźć szukany element (żądaną wartość). Oczywiście, żeby nasze wyszukiwanie binarne przebiegło pomyślnie, musimy na samym początku posortować najpierw naszą tablicę (jest bardzo ważna sprawa, ponieważ jak tego nie zrobimy, nasz rezultat nie będzie prawidłowy).

**Dokładny opis działania:**

Na samym początku nasza uporządkowana tablica jest dzielona na mniejsze przedziały do momentu, gdy nasz przedział osiągnie długość jeden, kiedy jednym sprawdzeniem można określić, czy element znajduje się w tablicy.

W pojedynczym kroku rozważa się jeden przedział charakteryzowany dwoma indeksami: **początkowym** oraz **końcowym**. Algorytm rozpoczyna wyszukiwanie od całej tablicy.

Kolejny etap to wyznaczenie środku przedziału:

**a** – indeks początkowy

**b** – indeks końcowy

Potem przedział jest zawężany – dzięki uporządkowaniu danych wiadomo, że albo poszukiwany element może znajdować się w indeksach [a, c] albo za nim. Inaczej mówiąc wybór ogranicza się do przedziału [a, c], gdy poszukiwany jest element mniejszy lub równy zapisanemu pod indeksem c lub [c + 1, b].

**Linki:**

* <https://pl.wikipedia.org/wiki/Wyszukiwanie_binarne>
* <http://strefakodera.pl/algorytmy/algorytmy-wyszukiwania/wyszukiwanie-binarne-algorytm-typu-dziel-i-zwyciezaj>

**16. Otwieranie i zamykanie plików, buforowanie.**

Co to jest buforowanie?

**Buforowanie** – jest to technologia polegająca na stosowaniu buforów programowych lub sprzętowych, jest ona najczęściej stosowana w celu wyrównania różnic prędkości przesyłania danych pomiędzy różnymi urządzeniami. Działa on zazwyczaj wykorzystując zasadę FIFO.

**Możemy wykorzystać bufory o**:

* Zerowej pojemności (bufor jest tylko przekaźnikiem komunikatorów)
* Ograniczonej pojemności
* Nieograniczonej pojemności

**Linki:**

* <https://pl.wikipedia.org/wiki/Buforowanie>

**17. Co to jest złożoność obliczeniowa? Jakie są klasy złożoności algorytmów? Przykłady notacji dużego ‘O’**

**Złożoność obliczeniowa** - jest to dziedzina informatyki teoretycznej, która zajmuje się klasyfikacją języków formalnych ze względu na ilość zasobu — **czasu i pamięci** — potrzebnego do rozpoznawania języka. Ponieważ język formalny jest abstrakcyjnym odpowiednikiem problemu algorytmicznego, teoria ta dostarcza narzędzi do szacowania trudności obliczeniowej takich problemów. Niniejszy kurs zapoznaje uczestników z najważniejszymi rezultatami badań dotyczącymi klas złożoności, ich własności i wzajemnych zależności a także omawia wynikające z tych faktów wnioski dotyczące praktycznych problemów algorytmicznych.

**Jakie są klasy złożoności algorytmów?**

* **Klasa problemów (P)** – są to głównie problemy łatwe, dla których istnieje rozwiązanie o złożoności wielomianowej lub niższej.
* **Klasa problemów (NP)** – są to problemy, dla których nie znamy rozwiązań w czasie wielomianowym lub mniejszym.
* Problemy **NP-zupełne (NPC)** – są to problemy klasy NP równoważne sobie w sensie rozwiązania – to znaczy takie, że gdyby udało się wielomianowo rozwiązać jeden – udałoby się rozwiązać wszystkie. Problemów tej klasy jest wiele i jest to większość konkretnych, życiowych problemów przed którymi stoi informatyka.
* Problemy **NP-trudne (NPD)** – znaleziono rozwiązania wyższe niż wielomianowe i zarazem dowiedziono braku rozwiązań wielomianowych.

**Notacja dużego O** – notacja przedstawiająca **asymptotyczne tempo wzrostu**, wykorzystywana do zapisywania złożoności obliczeniowej algorytmu. Za pomocą tej notacji zapisywany jest rząd wielkości funkcji wyrażającej liczbę operacji dominujących (w przypadku złożoności czasowej) lub rozmiar wymaganej pamięci (w przypadku złożoności pamięciowej) w zależności od liczby danych wejściowych.

**Przykłady**:

* f(n) = n2 + 100n + log10 n + 1000 możemy przybliżyć jako: f(n) ≈ n2 + 100n + O(log10 n) albo jako: f(n) ≈ O(n2).
* ½ n2 - 3n = O(n2), ale również np. 5n +6 = O(n2).

**Linki:**

* <https://www.samouczekprogramisty.pl/podstawy-zlozonosci-obliczeniowej/>
* <http://strefakodera.pl/algorytmy/inne/zlozonosc-obliczeniowa-co-to-takiego>
* <https://pl.wikipedia.org/wiki/Z%C5%82o%C5%BCono%C5%9B%C4%87_obliczeniowa>

**18. Kryteria oceny programów. Etapy programowania. Na czym polega uruchamianie a na czym testowanie?**

**Kryteria oceny programów:**

**Etapy programowania:**

* **Planowanie** – chodzi tutaj o sformułowanie problemu do rozwiązania, w tym etapie określamy cele projektu i zadania, które nasz program będzie realizować.
* **Tworzenie programu** – jest to etap, w którym przystępujemy do pisania programu, tworzymy kod źródłowy..
* **Kompilacja** – ten etap pozwala nam na sprawdzenie poprawności semantycznej i syntaktycznej programu oraz ustali rodzaj i lokalizację błędów.
* **Konsolidacja** – w procesie konsolidacji moduły programu, które powstały podczas kompilacji, są łączone w jeden program.
* **Testowanie** – testowanie programu to działanie zapewniające jakość oprogramowania. Ten etap jest stosowany w celu weryfikacji oraz walidacji oprogramowania.
* **Optymalizacja** – jest to działanie, którego celem jest poprawienie wydajności programu.

**Na czym polega uruchamianie, a na czym testowanie?**

**Uruchamianie** polega ono głównie na wykonaniu programu.

**Testowanie** polega głównie na uruchamianiu programu lub aplikacji z zamiarem znalezienia błędów oprogramowania. Inaczej mówiąc polega ono na walidacji oraz weryfikacji, czy program działa zgodnie z zamierzeniem, czy może zostać uruchomiony mając niezmienioną charakterystykę.

**Linki:**

* <http://www.informatyka.orawskie.pl/?pl_narzedzia-programistyczne-etapy-tworzenia-programu,230>

**19. Abstrakcyjne struktury i typy danych i ich implementacja**

Co to jest abstrakcyjna struktura danych?

**Abstrakcyjna struktura danych** – (dość ważne pojęcie w programowaniu i projektowaniu algorytmów), głównie jest to zbiór danych elementarnych wraz z dobrze zdefiniowanym na nich zbiorem operacji. Inaczej mówiąc abstrakcyjna struktura danych jest zdefiniowana przez tzw. zestaw operacji, które można wykonać na danych, przy tym nie wnikając w ich sposób implementacji.

**Linki:**

* <http://www.wozna.org/students/2014-2015/asd/asd05.pdf>
* <https://pl.wikipedia.org/wiki/Abstrakcyjny_typ_danych>

**20. Obiekty, klasy i metody. Klasy opakowujące. Konstruktory**.

**Obiekt** – jest to reprezentacja wartości określonego typu w pamięci komputera, charakteryzowana przez jego stan (wartość typu danych), zachowanie (operacje na typie danych) oraz tożsamość (lokalizację w pamięci).

**Metoda** – jest to nazwana sekwencja instrukcji, która może być wywołana przez inny kod w celu wykonania obliczeń.

**Klasa** – jest to konstrukcja języka, która służy do implementacji zdefiniowanego przez użytkownika typu danych, stanowiącego szablon do tworzenia obiektów zawierających wartości tego typu i manipulowania nimi.

**Klasy opakowujące**:

**Klasy osłonowe (opakowujące)** - są one jakby pomostem pomiędzy typami prostymi a obiektowymi. Ponieważ na typach prostych nie można wywoływać żadnych metod, jak robimy to na obiektach, wprowadzono klasy opakowujące, rozszerzające ich standardowe możliwości. Przykładowo obiekt typu Double reprezentujący liczby zmiennoprzecinkowe, można łatwo przekonwertować do typu long lub zamienić na string.

**Klasy osłonowe (opakowujące)** powstały dla wszystkich typów prostych:

* long
* int
* short
* byte
* char
* double
* float
* boolean

**Konstruktor** – jest to specjalna metoda danej klasy, która jest wywoływana podczas tworzenia jej instancji. Podstawowym zadaniem konstruktora jest zainicjowanie obiektu, a w niektórych językach programowania także utworzenie obiektu.

**Zadania konstruktora:**

* obliczanie rozmiaru obiektu.
* alokacja obiektu w pamięci.
* wykonanie kodu wywołanego konstruktora.
* wykonanie kodu klasy bazowej.
* wyczyszczenie (czyli zerowanie) obszaru pamięci zarezerwowanej dla obiektu.

**Linki:**

* <https://pl.wikipedia.org/wiki/Konstruktor_(programowanie_obiektowe)>
* <https://stormit.pl/klasy-oslonowe/>